FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Gestão otimizada de uma infraestrutura de switching redundante

João Correia

WORKING VERSION

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: João Neves

October 30, 2015

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES | **2014/2015**

A Dissertação intitulada

"Gestão Otimizada de uma Infraestrutura de Switching Redundante"

foi aprovada em provas realizadas em 16-10-2015

o júri

Presidente Professor Doutor Ricardo Santos Morla
Professor Auxiliar do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professor Doutor Jorge Botelho da Costa Mamede
Professor Adjunto do Departamento de Engenharia Eletrotécnica Instituto Superior de Engenharia do Porto

Professor Doutor João Manuel Couto das Neves
Professor Auxiliar Convidado do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projeto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extratos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são corretamente citados.

Autor - João Gonçalo Guimarães Correia

Faculdade de Engenharia da Universidade do Porto

Resumo

Com o desenvolvimento das redes de computadores, bem como, o aumento da nossa dependência perante elas, torna-se imperativo aumentar a fiabilidade das mesmas, de forma a reduzir a perda de conectividade quando um problema acontece. Uma falha de conectividade de longa duração pode provocar perdas monetárias ou colocar pessoas e bens em perigo.

Temos neste caso de recorrer a técnicas de redundância para assegurar os serviços necessários. Redundância numa rede de computadores é quando introduzimos equipamentos extras na rede de forma a garantir que quando algum problema ocorre existem caminhos adicionais para manter a conexão. Geralmente temos para este efeito uma duplicação de equipamentos nas camadas principais de uma rede assim como um aumento das ligações entre os componentes.

Na presença destes equipamentos extras, a nossa rede fica mais complexa e mais difícil de gerir, é então necessário introduzir maneiras de controlar o aumento dos componentes de forma a gerir-los. Para isso usamos protocolos como *Spanning Tree* e protocolos de redundância para impedir a criação de *loops* na nossa rede.

Este projeto de dissertação nasce então da necessidade de estudar as tecnologias que nos permitam gerir e controlar a redundância introduzida numa rede assim como descobrir novas maneiras de o fazer. Tendo este objetivo em mente este projeto foca-se em encontrar uma solução ótima e implementa-la de forma a testar a sua resiliência.

É também pertinente o estudo de técnicas para otimizar a gestão de uma rede deste calibre de forma a diminuir a dificuldade e o tempo necessário que o processo de gestão toma, assim como o estudo de serviços a disponibilizar numa rede com estas características.

Abstract

With the development of computer networks and the increasing dependence on them, we need more reliability to ensure a reduction of loss of connectivity when a problem arises. A long loss of connection may cause negative effects on a business and may endanger lives or property.

In this case we need to use redundancy techniques to ensure the necessary services. Network redundancy occurs when we introduce additional equipments to the network to make sure that if a problem occurs, additional paths exist to maintain connection. Normally for this effect we have a duplication of the key network components and an increase of links between them.

In the presence of this additional equipments, our network becomes more complex and difficult to manage, we need to find a way to control the increase of components so that we can manage it. To do so we use protocols like Spanning Tree and Redundancy protocols to eliminate loops in our network.

This dissertation project came from the necessity to study and research technologies that allows us to manage and control the introduced redundancy as well as to find additional ways to do so. With this goal in mind this project focus on finding an optimal solution and implement it so that we can test its resilience.

It is also relevant to study the techniques that optimize the management of a network of this scale, in order to make this process easier and faster, as well as, the study of services that can be provided on a network with this characteristics.

Acknowledgements

This project concludes one of the most important stages of my academic journey, the hardships encountered were surpassed with the support of dear ones and I would like to thank them.

In the first place, I would like to thank my close friends and colleagues that walked this path alongside me and helped me during the course of this project. From this circle of friends I would like to specially thank Carlos Leocádio, Carlos Pina, Hugo Ribeiro, Madalena Ferreira, Pedro Araujo, Pedro Nogueira and Pedro Pires for all the help that they offered and all the motivation that they gave me.

I would also like to thank my parents and sister for the opportunity that they provided me and for all the support during all of my academic life.

I thank my supervisor, João Neves, for proposing this project and for all the guidance that he gave me, allowing me to finish this project that taught me so much.

Lastly I would like to thank Paulo Vaz for the help he gave me, most of the times when I was on a dead end.

Thank you.

João Correia

"The first rule of any technology used in a business is that automation applied to an efficient operation will magnify the efficiency. The second is that automation applied to an inefficient operation will magnify the inefficiency."

Bill Gates

Contents

List of Figures

List of Tables

List of Listings

Abbreviations, Acronyms and Initials

API	Application programming interface
ARP	Address Resolution Protocol
AS	Autonomous System
AVF	Active Virtual Forwarder
AVG	Active Virtual Gateway
BGP	Border Gateway Protocol
BPDU	Bridge Protocol Data Units
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
EC	EthernetChannel
ECMP	Equal Cost Multi-Path Routing
EIGRP	Enhanced Interior Gateway Routing Protocol
FEUP	Faculdade de Engenharia da Universidade do Porto
FHRP	First Hop Redundancy Protocol
FIB	Forwarding information base
FTP	File Transfer Protocol
GLBP	Gateway Load Balancing Protocol
HSRP	Hot Standby Router Protocol
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
INESC TEC	INstituto de Engenharia de Sistemas e Computadores - Tecnologia e Ciência
IP	Internet Protocol
ISP	Internet Service Provider
LACP	Link Aggregation Control Protocol
LAN	Local Area Network
MAC	Media Access Control
MEC	Multichassis EtherChannel
MRTG	Multi Router Traffic Grapher
NAT	Network Address Translation
NSF	Nonstop Forwarding
NTP	Network Time Protocol
OSI	Open Systems Interconnection
OSPF	Open Shortest Path First
PAgP	Port Aggregation Protocol
PVST	Per-VLAN Spanning Tree
RFC	Request for Comments

RP	Routing Protocol
RSTP	Rapid Spanning Tree Protocol
SDN	Software Defined Networking
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SSH	Secure Shell
SSO	Stateful Switchover
STP	Spanning Tree Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VLAN	Virtual LAN
VOIP	Voice over IP
VPN	Virtual Private Network
VRRP	Virtual Router Redundancy Protocol
VSL	Virtual Switch Link
VSS	Virtual Switching System

Chapter 1

Introduction

This chapter introduces the context, motivation and problem of this dissertation project, providing a global overview of its structure as well.

1.1 Context

This thesis project was proposed by Professor João Neves and it falls under the programme of the Master's Degree in Electrical and Computer Engineering of the Faculdade de Engenharia da Universidade do Porto (FEUP).

1.2 Motivation

With the development of data networks, the world started to rely more on them. We use them in our homes, companies, hospitals, almost everywhere. Because of this we need them to be reliable, available and easy to manage, since a simple 2 minutes down time could cause a loss of millions of Euro to some companies or in case of a medical facility it could endanger human lives.

To help prevent such unwanted sceneries, there are certain configurations and protocols that can be implemented on a network, normally known as adding redundancy to a network. This allows the creation of additional paths in the network that can be used when a disorder occurs, be it a device failure or a broken link. Redundancy in some cases can also increase the bandwidth of certain paths improving performance.

1.3 Purpose and Objectives

The problem of this project is the study of the balance between network redundancy and its increased management complexity. With this problem in mind the objectives for this project are the following:

- Study of redundancy solutions for a enterprise sized network.

- Establishment of a reliable enterprise sized network.
- Study of services to provide on the network.

The desired network resembles the one presented on figure 1.1, consisting of a core layer, composed of two switches, supporting a distribution layer that connects to the access layer, that provides connectivity to the user terminals and the rest of the equipments.

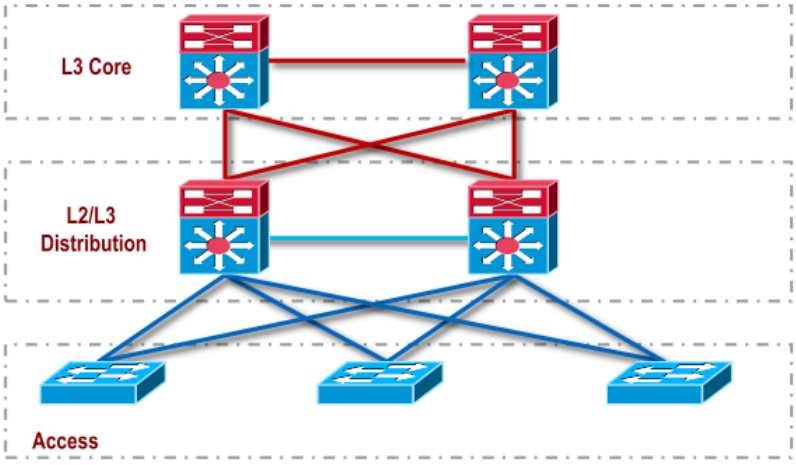


Figure 1.1: A L2/L3 network[1, p.8]

1.4 Document Structure

This document is arranged in 6 chapters.

As stated before, this chapter describes the context, motivation of this project and its objectives.

The second chapter begins with a description of network redundancy and management and then presents technologies and protocols that can be used to increase network redundancy and simplify its management.

The third chapter describes some of the possible solutions to the problem at hand.

The forth chapter describes the implementation of the chosen solution.

The fifth chapter describes the tests performed to validate the solution implemented and to assert the reliability of the network.

The sixth chapter is a brief conclusion.

Lastly we have the appendixes, in appendix one the necessary configurations (to activate Cisco Virtual Switching System and for the implementation) are presented and in appendix two we have some of the python scripts used.

Chapter 2

State of the Art

This chapter will introduce some important concepts and some of the possible protocols and technologies to help solve this problem, focusing on those using Cisco equipment since it is the one available for this project.

2.1 Important Concepts

In order to properly find a solution some important concepts are needed. In this first section the concepts of network redundancy and management are introduced so that proper solutions can be researched.

2.1.1 Network Redundancy

Network redundancy is the installation of additional or alternate instances of network devices, equipment and communication mediums. This process helps to ensure network availability in case of a network device or path failure and unavailability. As such, it provides a means of network failover.

Network redundancy is primarily implemented in enterprise network infrastructure to provide a redundant source of network communications. It serves as a backup mechanism for quickly swapping network operations onto redundant infrastructure in the event of unplanned network outages.

Typically, network designs achieve redundancy through the addition of alternate network paths, in this manner, through redundant standby router at the third layer of the Open Systems Interconnection (OSI) model and backup switches at layer 2. When the primary path is unavailable, the alternate path can be instantly deployed to ensure minimal downtime and continuity of network services.

2.1.2 Network Management

Network management is a broad range of activities, methods, procedures and the use of tools to administrate, operate, and reliably maintain computer network systems. Being such a broad concept, it is possible to divide network management into different areas: Administration, Operation, Maintenance and Provisioning.

Network Administration is the area that involves tracking the network resources such as transmission lines, hubs, switches, routers, and servers. It also involves monitoring their performance and updating their associated software.

Network Operation refers to the smooth network functioning as designed and intended, including close monitoring of activities to quickly and efficiently address and fix problems as they occur and preferably even before users are aware of the problem.

Network Maintenance involves timely repair and necessary upgrades to all network resources as well as preventive and corrective measures, like replacing or upgrading network equipment such as switches, routers or damaged transmission lines.

Lastly Network Provisioning refers to configuring network resources to support the requirements of a particular service.

2.2 Spanning Tree Protocol (STP)

Spanning Tree Protocol (STP) is a protocol used by network equipments to solve problems caused by loops in a bridged Ethernet Local Area Network (LAN). The purpose of STP is to avert loops in a bridged LAN and to eliminate broadcast storms and other problems that arise with loops. STP helps to manage the redundant links between switches on the second layer, providing control of automatic backup paths without the need to manually activate them. STP was standardized as IEEE 802.1d but disfavoured as of 2004 by the Rapid Spanning Tree Protocol (RSTP).

2.2.1 STP operation

In a Local area network (LAN), be it a single segment or an aggregation of bridges and LAN segments, the bridges collectively compute a spanning tree in order to eliminate loops in the topology while maintaining access to all the segments.

First a root bridge is selected, the bridge with the lowest bridge Identifier (ID). The bridge ID is composed of a configurable priority number, by default 32768 and the MAC address. The priority number can be modified by a network administrator to manually choose a bridge root to reduce the cost of the spanning tree.

After the root bridge is selected each bridge determines the least-cost path to the root. The port connecting to that path becomes the root port (RP). The bridges connected to each segment then compute the bridge that has the least-cost path from the segment to the root. The port connecting the selected bridge to the segment becomes the designated port (DP) for that segment.

All the active ports that are not a root port or a designated port become a blocked port (BP).

An example of a spanning tree is represented on Figure 2.1.

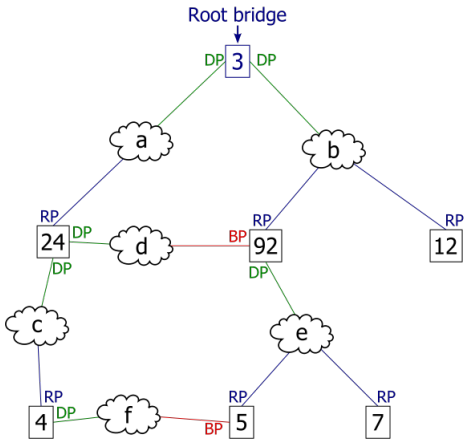


Figure 2.1: The STP at work. Numbers represent bridges and letters LAN segments.

In case of a link failure the spanning tree algorithm computes and spans a new tree. Figure 2.2 shows the new spanning tree computed when a link failure occurs in the tree of Figure 2.1.

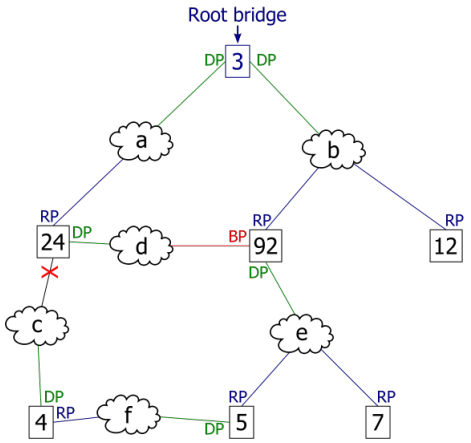


Figure 2.2: The STP at work after a link failure.

The process described above specifies how the bridges determine what spanning tree will be computed, but each bridge requires information about the other bridges to do so. To ensure that each bridge has enough information, the bridges use special data frames called Bridge Protocol Data Units (BPDU) to exchange the necessary information.

There are two types of BPDUs, first we have Configuration BPDU (CBPDU), that are used for the computation of the spanning tree, and Topology Change Notification (TCN) BPDU, used to announce changes in the network topology.

BPDUs are exchanged regularly, by default every 2 seconds, and are sent to the STP multicast address 01:80:C2:00:00:00. They enable the switches to keep track of network changes that lets them start or stop forwarding at each port as required.

2.2.2 Rapid Spanning Tree Protocol (RSTP)

In 2001, the Rapid Spanning Tree Protocol was introduced by the IEEE as 802.1w. RSTP presents a significant boost in the speed of the spanning tree convergence after a topology change, through new convergence behaviours and bridge port roles.

RSTP is much faster than normal STP, it usually takes 3 times the time of the specified HELLO time, which default value is 2 seconds, to respond to a change in the topology or in case of a link failure it recovers in a few milliseconds, where STP would normally take at least 30 seconds to do so. [2]

2.2.2.1 RSTP operation

In RSTP the bridges ports that would have been blocked in STP have two distinct roles that are Alternate Port (AP) and Backup Port (BP). An alternate port occurs when there is already a designated port of another bridge to a LAN segment, and receives more useful BPDUs from another bridge as is shown on figure 2.3. [2]

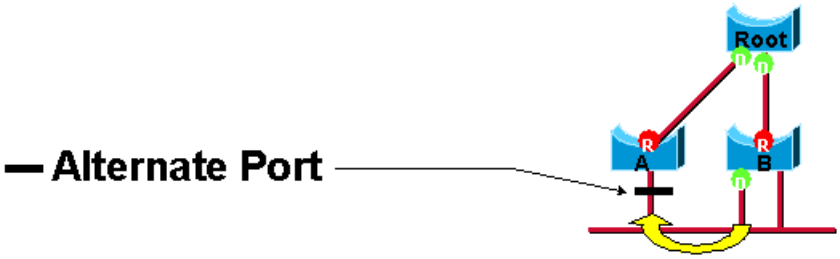


Figure 2.3: An Alternate Port.[2]

An Backup Port is a blocked port that connects a bridge to a segment that already has a designated port from the same bridge, and receives more useful BPDUs from the same bridge it is on as is shown on figure 2.4.[2]

These new ports roles allow RSTP to converge much faster, than standard STP, after a failure since it already has some of the information necessary to compute a new tree after a failure.

Figure 2.4: An Backup Port.[2]

2.3 First Hop Redundancy Protocols (FHRP)

While STP can be used to manage the redundancy at the second layer, but when redundancy needs to be implemented on the third layer normally a FHRP is used. FHRP protects the default gateway of a network, by allowing two or more routers to provide backup for the same address. In case of a failure, one of the backup routers can take the role of primary router and serve as the default gateway.

These protocols can be used to provide the same service to devices other than routers, like having backup servers to respond to requests, in case the main one fails.

2.3.1 Hot Standby Router Protocol (HSRP)

Hot Standby Router Protocol is a redundancy protocol developed by Cisco and is described in detail in RFC 2281 [3]. This protocol is used to provide redundancy to a default gateway making the network resilient to gateway failures that may arise.

By using this protocol the routers communicate between them establishing a virtual router that, for the rest of the network, behaves as a single entity. Normally HSRP works in association with a rapid-converging routing protocol like Enhanced Interior Gateway Routing Protocol (EIGRP) or Open Shortest Path First (OSPF).

The routers communicate by sending messages using a multicast address (224.0.0.2 for version 1 or 224.0.0.102 for version 2) and the UDP port 1985. By sending HELLO messages to other HSRP enabled routers, the routers define the priority between them.

The router with the highest configured priority is elected as the Active router, with a pre-defined gateway IP address, and is responsible for forwarding the packets that hosts send to the virtual router. The router with the next-highest priority is selected as the Standby router. In the event of failure of the active router, the standby router will assume the packet forwarding duties of the virtual router, thus achieving default gateway failover.[4][5]

2.3.2 Virtual Router Redundancy Protocol (VRRP)

Virtual Router Redundancy Protocol, which is described in IETF publication RFC 5798[6], is a FHRP that automatically assigns routers to hosts. In doing so it increases the availability and reliability of routing paths.

VRRP creates virtual routers that represent a group of routers. In each host the default gateway is assigned to a virtual router, composed of a master and backup routers. In case of failure of the master router of a given virtual router one of the backup routers takes its place and forwards the packets delivered to the virtual router.

The master router takes 00:00:5E:00:01:XX as its MAC address, the last byte of the address is the Virtual Router Identifier, which is unique to each virtual router. When a Address Resolution Protocol (ARP) request is sent for the virtual router IP, the master router replies with that MAC address. Routers participating on the same VRRP group send the VRRP HELLO messages to the multicast address 01:00:5E:00:00:12.[5]

When the master router fails to send a multicast packet, for a period longer than three times the advertisement timer, the backup routers assumes that the master router is dead. The virtual router then enters an unsteady state and the election process for the new master router begins.

Backup routers only send multicast packets during an election process or when a router is configured with a higher priority that the current master, which means that this router will take over the role of master router.[6]

2.3.3 Gateway Load Balancing Protocol (GLBP)

Gateway Load Balancing Protocol is another Cisco's proprietary FHRP, that introduces load balancing functionalities.

Like in other protocols of its kind, in GLBP each router has a configurable priority, however GLBP allows a weighting parameter to be set. Using this new parameter GLBP is able to balance the load of the routers by enabling ARP requests to be answered by different routers.

Like in VRRP the routers work in groups to provide gateway redundancy. For each group it is selected an Active Virtual Gateway (AVG), them if the group has more than two members, the second best AVG is placed in Standby state and all the others routers are placed in the Listening state. The AVG then assigns a virtual MAC address to each member of the group (00:07:B4:00:XX:YY where XX is the GLBP group and YY the member of the group), including itself , enabling Active Virtual Forwards (AVFs). Each group can have four AVFs at the same time.

When an ARP request is sent to the virtual IP the AVG responds with the virtual mac of one of the members depending on the load balancing scheme configured. There are three different modes for load balancing, first we have Round-Robin in which the AVG responds to ARP requests with the same AVF during a specific set of time, after this time passes it them proceeds to the next AVF. Then we have Weight mode that uses the weight parameter of each AVF to balance the load of the members according to the configured value. Lastly we have the host defined mode, in which every time an host sends an ARP request the AVG always responds with the same AVF.

By default, GLBP routers use the local multicast address 224.0.0.102 to send HELLO packets to their peers every 3 seconds over the UDP port 3222.

2.3.4 Comparison of FHRPs

These three protocols are quite similar since they all serve the same purpose and have similar characteristics, like IPv6 support. GLBP is superior since it provides load balancing and an increased number of maximum groups of clients but at the cost of performance. Table 2.1 highlights this comparison of the protocols.

	HSRPv2	**VRRPv3**	**GLBP**
IPv6 Support	yes	yes	yes
Load Balancing	no	no	yes
Maximum Groups/interfaces	255	255	1024
Name syntax	Active, Standby	Master, Backup	AVG, AVF
Multicast addr. IPv4	224.0.0.102	224.0.0.18	224.0.0.102
Virtual Mac addr	00:00:0E:07:AC:XX	00:00:5E:00:01:XX	Added from AVG

Table 2.1: Redundancy protocols comparison[5]

It is possible to provide load balancing to HSRP and VRRP with the configuration of various groups instead of only one per subnetwork, but this makes the network harder to manage since all the hosts need to be configured with a virtual IP manually.

In terms of performance all three protocols are quite similar, being HSRP the faster to converge followed by GLBP and VRRP respectively.[5]

2.4 Routing Protocols

A routing protocol serves the purpose of determining the appropriate path over which data is transmitted between two network nodes, specifying how routers share information with other router of the same network.

Routing Protocols often save more than one path to the same network but only present the best one, with this it is possible to control the redundancy of a network, since the network nodes may know more than one path to the same network even if one of the paths becomes unavailable others paths remain in its routing table.

2.4.1 Border Gateway Protocol (BGP)

Border Gateway Protocol, described in IETF publication RFC 4271 [7], routes traffic between autonomous systems. An autonomous system is a network or group of networks under the same administration and with common routing policies. BGP exchanges routing information for the Internet and is the protocol used between ISPs. ISPs use BGP to exchange customer and ISP routes. When BGP is used between autonomous systems, the protocol is referred to as external BGP (eBGP). If a service provider is using BGP to exchange routes within an autonomous system, the protocol is referred to as interior BGP (iBGP).

BGP is a very robust and scalable routing protocol, as evidenced by the fact that it is the routing protocol employed on the Internet. To achieve scalability at this level, BGP uses many route parameters, to define routing policies and maintain a stable routing environment. BGP neighbours exchange full routing information when the TCP connection between neighbours is first established. When changes to the routing table are detected, the BGP routers send to their neighbours only those routes that have changed. BGP routers do not send periodic routing updates, and BGP routing updates advertise only the optimal path to a destination network.

2.5 Link Aggregation

Link Aggregation refers to various methods of combining multiple physical connections in parallel to increase the throughput of a point-to-point connection and/or to provide link redundancy. There are several standards when it comes to Link Aggregation, one of them is vendor independent like the Link Aggregation Control Protocol, but there are proprietary solutions, for example EtherChannel from Cisco.

Link aggregation can work in various modes, having only one link active while the others stay as backups, rotate the active link so no link is used more that a certain time or using load balancing to divide the traffic between all the links aggregated which improves the throughput to almost the number of links aggregated times their speed. In the case of 8 aggregated links which is the limit of aggregated links, we can have connections up to 800Mbits/s in case of 10Mbits/s links, 8Gbit/s in case of 1Gbits/s links or 80Gbits/s in case of 10Gbits/s links.

So far the only solution for multichassis link aggregation is the Multichassis EtherChannel developed by Cisco and only available on Cisco's Virtual Switching System.

2.6 Stackable Switches

A stackable switch works like any other switch when operating on standalone, but can be set to operate together with one or more switches, becoming a stack. During the initial configuration of a stack it receives an IP address for the whole stack which simplifies the management of the switches, since the system administrator can control the whole stack from a single connection.

Stackable switches used in combination with link aggregation can be used to improve the resilience of the network since they permit the aggregation of links connected in different switches of the same stack.[8]

Even if one of the switches fails the the rest of the stack can still work properly and with link aggregation can provide the necessary connection to other levels of the network.

2.7 Virtual Switching System (VSS)

Cisco's Virtual Switching System permits the merge of two or more physical switches into an unique logical entity. This technology grants improvements in availability, reliability, management, scalability and maintenance.

The Virtual Switching System is created by logically merging two standalone Cisco Catalyst able systems. This conversion is a one-time process that requires simple configuration steps.

The VSS can be seen as a single logical network entity from the network control plane and management perspectives. To neighbouring devices, the VSS appears as a single logical switch or router, due to the use of Cisco IOS Stateful Switchover technology, as well as Non-Stop Forwarding extensions to routing protocols.[9]

2.7.1 VSS Architecture

Within a VSS, one chassis is designated as the active virtual switch while the other is the standby virtual switch. The control plane functions are managed by the active supervisor engine of the active virtual switch, including:

- Management (Telnet, SSH, SNMP, etc)
- Layer 2 protocols (STP, LACP, etc)
- Layer 3 protocols (routing protocols, etc)
- Software data path

From the data plane perspective both switches in the VSS actively forward traffic but only the active virtual switch has the control plane active making him responsible for the management of both equipments. This is better visualized on figure 2.5.

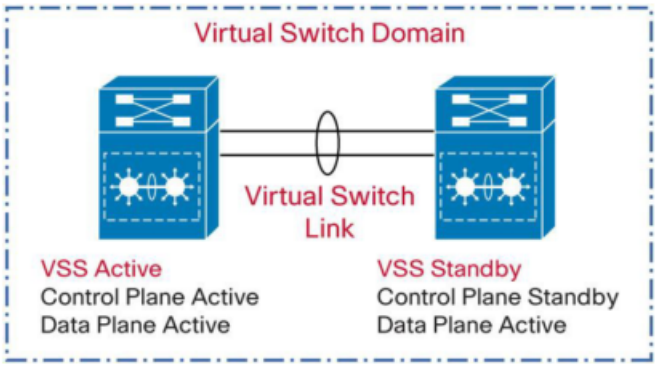


Figure 2.5: Components of a VSS.[9]

2.7.2 Virtual Switching Link (VSL)

Since the VSS consists of two chassis, in order to bond them into a single logical node, special signalling and control information must be exchanged. To facilitate this information exchange, a dedicated link is used to transfer both data and control traffic between the two chassis. This link is referred to as the virtual switch link.

The VSL is formed as a EtherChannel interface and can comprise of one to eight physical member ports.

All the communication over the VSL is encapsulated with a virtual switch header, which is appended to the frames. This header is placed after the Ethernet preamble and directly before the layer 2 header as seen on figure 2.6.

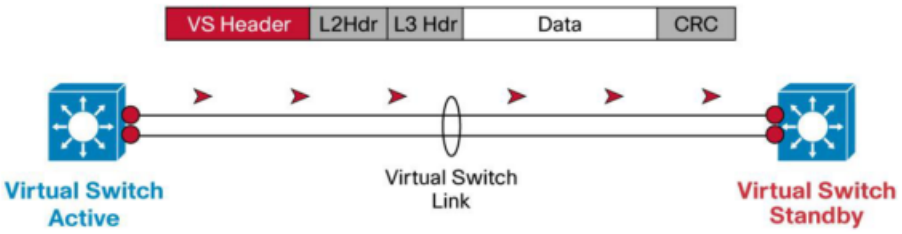


Figure 2.6: VS Header.[9]

2.7.3 High Availability

VSS can provide High Availability like other solutions but it is simpler to implement. VSS takes care of all the management of the connection between the two redundant switches and automatically bonds the links from the other levels of the network using Cisco's proprietary solution MultiChassis EtherChannels (MECs).

2.7.4 Hardware and Software requirements

VSS is supported on Catalyst 6500, 4500-E and 4500-X but require specifics supervisors. For Catalyst 6500 the supervisors Sup2T or Sup720-10G are required but the Catalyst 4500-E require supervisors Sup7-E or Sup7L-E for it to be possible to create a VSS.

For VSS created from Catalyst 6500 and 4500-E, there is not a need for the chassis and the installed modules to be the same, this means that we can create a VSS using a Catalyst 6506 and a Catalyst 6509 that have different modules installed.

2.8 Software Defined Networking (SDN)

SDN is the physical separation of the network control plane from the forwarding plane enabling one control plane to control several devices[10].

SDN is an emerging architecture, that is dynamic and easily manageable. Today switches and routers already have their control and data planes decoupled even through they run on the same chassis, this architecture further decouples them and allow the control plane to be moved outside of the chassis to a centralized controller.[11]

This controller dictates the overall network behaviour, making it easier to introduce changes in the behaviour through a software program that manually configure several equipments with a set of commands. This simplifies the management of a network making it easier to administrators to adapt the network to changes as needed.[12]

To be possible to implement SDN, the network equipments need to be able to communicate with the controller, normally called the southbound SDN interface. The most common one is OpenFlow since several vendors produce OpenFlow capable equipment. the Open Networking Foundation is responsible for standardizing the OpenFlow protocol. There are several controllers with compatibility with OpenFlow, for example NOX, Floodlight and Maestro. Nox is a controller that facilitates the development of C++ and Python controllers, Floodlight is a java-based controller and Maestro focus on achieving better performance using multithreading. This architecture is shown on figure 2.7.

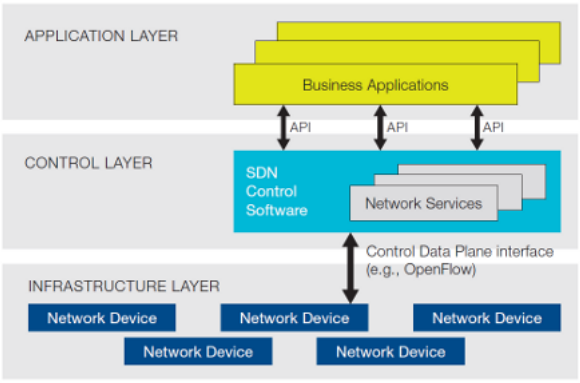


Figure 2.7: SDN architecture.

2.9 OpenStack

OpenStack is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a datacenter, all managed through a dashboard. It is backed by some of the biggest companies in software development and hosting as well as individuals community members. It is managed by the OpenStack Foundation.[13]

This system gives its users the ability to deploy virtual machines and other instances that can handle different tasks for managing a cloud environment quite easily.

OpenStack is open source software, that means that anyone who needs to make modifications to adapt OpenStack to its needs can access the source code and do so, and freely share those changes to the community.[14]

2.9.1 Key components

OpenStack is made up by several components, given its open nature anyone can add additional components to OpenStack to help it to meet their needs. However the OpenStack community has collaboratively identified nine key components that are distributed as a part of any OpenStack system and officially maintained by the OpenStack community.

2.9.1.1 Nova - Computing

Nova is the primary computing engine behind OpenStack. It is a "fabric controller" which is used for deploying and managing large numbers of virtual machines and other instances to handle computing tasks.

Nova can work with widely available virtualization technologies. KVM, VMware, and Xen are available choices for hypervisor technology, together with Hyper-V and Linux container technology such as LXC

2.9.1.2 Swift - Object Storage

Swift is a storage system for objects and files. Rather than the traditional idea of a referring to files by their location on a disk drive, developers can instead refer to a unique identifier referring to the file or piece of information and let OpenStack decide where to store this information.

Swift is redundant system responsible for ensuring data replication and integrity across the cluster. In the event of a server or hard drive failure, Swift replicates their content from other active nodes to new locations in the cluster.

2.9.1.3 Cinder - Block Storage

Cinder is a block storage component, which is more analogous to the traditional notion of a computer being able to access specific locations on a disk drive.

Cinder manages the creation, attaching and detaching of the block devices to servers. Block storage volumes are fully integrated into Nova and the Dashboard, allowing for cloud users to manage their own storage needs.

2.9.1.4 Neutron - Networking

Neutron provides the networking capability for OpenStack. It helps to ensure that each of the components of an OpenStack deployment can communicate with each other quickly and efficiently.

Neutron provides networking models for different applications or user groups. It manages IP addresses, allowing for dedicated static IP addresses or DHCP. Floating IP addresses let traffic be dynamically rerouted to any resources in the IT infrastructure, so users can redirect traffic during maintenance or in case of a failure.

Users can create their own networks, control traffic, and connect servers and devices to one or more networks. Administrators can use software-defined networking (SDN) technology like

OpenFlow to support high levels of multi-tenancy and massive scale. Neutron provides an extension framework that can deploy and manage additional network services such as intrusion detection systems (IDS), load balancing, firewalls, and virtual private networks (VPN).

2.9.1.5 Horizon - Dashboard

Horizon is the dashboard behind OpenStack. It is the only graphical interface to OpenStack, so for users wanting to give OpenStack a try, this may be the first component they actually "see" Developers can access all of the components of OpenStack individually through an application programming interface (API), but the dashboard provides to the system administrators a look at what is going on in the cloud and allow them to manage it as needed.

2.9.1.6 Keystone - Identity Service

Keystone provides identity services for OpenStack. It is essentially a central list of all of the users of the OpenStack cloud mapped against all of the services provided by the cloud and the users permission. It provides multiple means of access, meaning developers can easily map their existing user access methods against Keystone.

2.9.1.7 Glance - Image Service

Glance provides image (virtual copies of hard disks) services to OpenStack. Glance allows these images to be used as templates when deploying new virtual machine instances.

2.9.1.8 Ceilometer - Telemetry

Ceilometer provides telemetry services, which allow the cloud to provide billing services to individual users of the cloud. It also keeps a verifiable count of each user's system usage of each of the various components of an OpenStack cloud.

2.9.1.9 Heat - Orchestration

Heat is the orchestration component of OpenStack, which allows developers to store the requirements of a cloud application in a file that defines what resources are necessary for that application. In this way, it helps to manage the infrastructure needed for a cloud service to run.

2.10 Python

Python is an interpreted, interactive, object-oriented programming language. It combines remarkable power with very clear syntax and has many interfaces with system calls and libraries. Python is a high-level general purpose language that can be applied to many different problems, be it simple scripts to boost productivity or more complex applications capable of working with internet

protocols such as HTTP, FTP and SMTP or operating directly on the system interfaces like file systems or TCP/IP sockets [15].

Python has a wide variety of third-party extensions like Paramiko an implementation of the SSH2 protocol that let us send commands though a SSH session[16] or ciscoconfparse a library which parses through Cisco IOS style configurations.

Using some of the available libraries we can create scripts to automate equipment configurations and or monitor our network. For this effect exists a library called Netmiko that enhances the library previously refereed Paramiko, providing us with functions to connect to a network equipment through SSH and send commands to configure or view information about the connected equipment.

Other libraries enhance the Simple Network Management Protocol (SNMP) functionalities of Python letting us send SNMP requests to get information on the equipment or to alter its configuration.

Chapter 3

Solutions

This chapter introduces the possible solution to establish a reliable enterprise sized network and a brief comparison between them.

3.1 Problem Analyses

This project introduces some of the main problems around networks in this era. With the necessity for reliable and available connection to content like servers or data storages, it is required to implement certain precautions to ensure that the levels of performance and availability required are met.

These requirements bring us several problems, like how to ensure the redundancy of a key network device or a specific link. Generally this is achieved by duplicating the desired device and its connections and adding the relevant connections between them. For example a core switch should be duplicated so that even if one of the switches fails there is always one active to provide connectivity, as shown on figure 3.1.

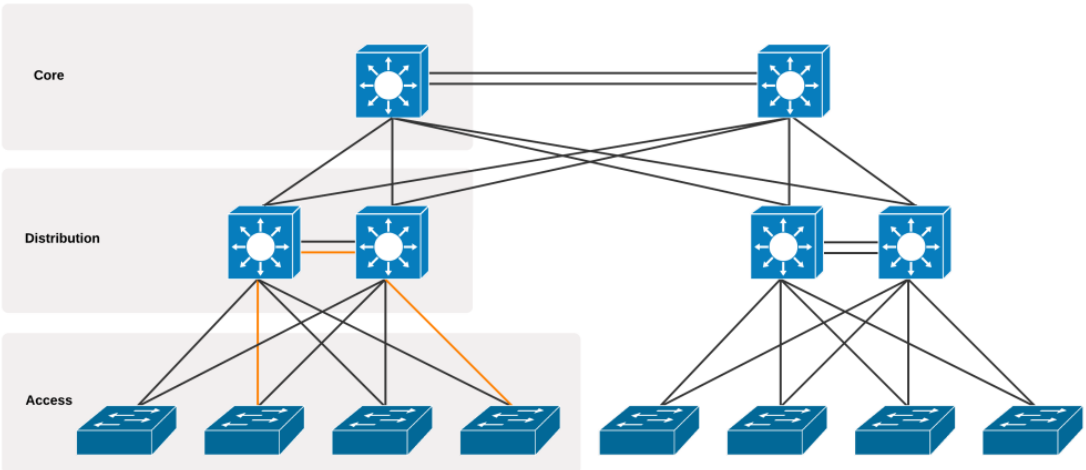


Figure 3.1: Network equipment duplication.

However this brings another problem that is the difficulty to manage the network with all the additional components. This requires certain protocols and/or technologies that smooth the management and control of a more complex network.

3.2 Possible Solutions

There are several ways to add and manage the redundancy of a network. We describe two different solutions on this chapter: one solution that works with equipment from any manufacturer and one another that relies on Cisco's proprietary technology.

3.2.1 General Solution

First we have the general solution. This solution is manufacturer independent, it uses standard protocols and does not require any technology available to a specific company. Those conditions make this the most used solution, easier to implement, due to the vast amount of information available.

This solution manages the redundancy of the network using some of the protocols and technologies referred in the previous chapter such as: Spanning Tree, Redundancy and Routing Protocols and Link aggregation. This technologies combined with the correct duplication of the key network equipments, enable the creation of a reliable and stable network that can sustain equipment and link failures.

Considering an enterprise sized network, we will have a duplication of the Core and Distribution switches and/or routers. When we do this we require the protocols mentioned before to manage the increased number of equipment and links, so we use Spanning Tree Protocol to ensure that we do not end up with loops in our network.

However in order to reach connectivity with the outside we need another layer of equipment above the core switches, normally we have at least one border router connected to the core layer, this requires us to manage how they communicate with each other. In order to manage the default gateway of the hosts we set up a Redundancy Protocol on the core layer so that we distribute a single IP address to the hosts. But we still have the problem of how the core layers communicates with the border routers, at this point a routing protocol is required to distribute the routes of the border routers to the core layer so that when a request is sent from a host and one of the core equipments receives that request, it knows where to forward that request to.

Another precaution we can take is to make use of the link aggregation technology to ensure that the links between our equipment are reliable, not only from the point of view of performance but also from the point of view of failover.

In general this will be our setup for a possible solution, as we can see in more detail in figure 3.2, we have STP on the Distribution and Core layers, a FHRP on the Core layer, Link aggregation between our equipment and a Routing protocol running between the Core layer and the Border routers.

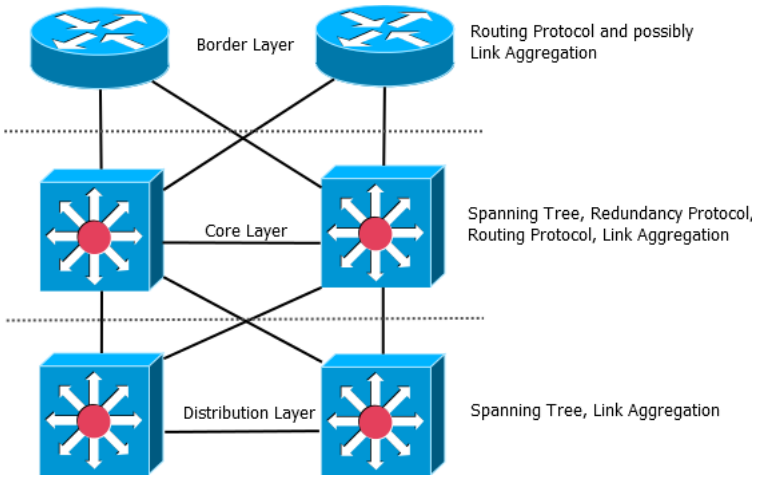


Figure 3.2: Protocols per Layer.

3.2.1.1 Advantages and disadvantages

The main advantage of this solution is that it works with any manufacturer's equipment, making it possible to implement in almost every scenery that we may encounter. Other advantage is that its quite simple to implement and we can encounter a lot of information about the technologies involved on the internet or in books.

The main disadvantage is that this solution may be difficult to manage given the size that the network may have. Certain protocols used on this solution were not designed with the goal of redundancy in mind, like the case of STP that was designed to prevent loops, this makes this solution not optimized as we desire, given that the protocols may take more time that we want to recover.

3.2.2 Virtual Switching System

Virtual Switching System Cisco's Proprietary solution to simplify the management of a redundant network. This technology was created with the goal of optimizing the behaviour of a network where redundancy is desired. The VSS enables the merge of two Cisco Catalyst 6500 or 4500 series switches, making it seem like a single logical unit.

This allow us to remove some of the protocols needed on the previous solution such as the FHRP used on the core layer. VSS takes care of much of the management of the corresponding layer. It optimizes the management of link aggregation by enabling the creation of Multichassis Etherchannels which opens new possibilities.

VSS also reduces the use of Spanning Tree since some of the links that were blocked by it are now used to create MECs that use them to improve the throughput.

Figure 4.2 presents us with the characteristics of a network with the core and distribution layer composed of a VSS.

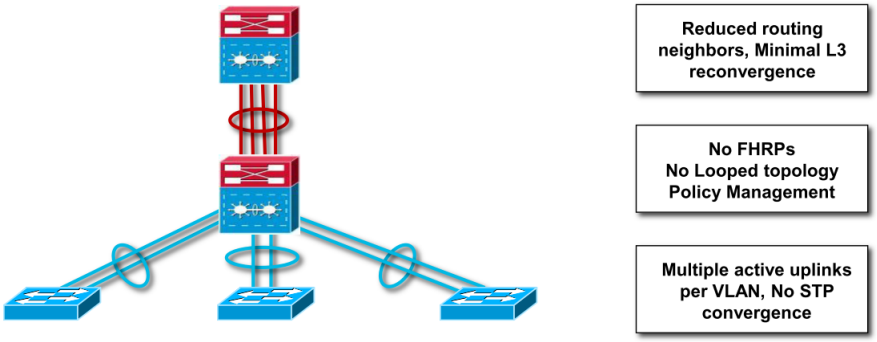


Figure 3.3: VSS network characteristic [1]

3.2.2.1 Advantages and disadvantages

VSS brings to the table several advantages, we have a reduction of the presence of STP, elimination of the redundancy protocol, better throughput and less routing neighbours. VSS also simplifies the management of the network since the two merged chassis have only one configuration file that covers the two chassis. The creation of MECs improve the link aggregation and lets us use some of the links that were blocked by STP [19].

Of course that this solution also has some disadvantages, such as the fact that is a Cisco proprietary solution that only works with some of Cisco's chassis, Catalyst 6500 series and Catalyst 4500 series.

3.3 Choosing a solution

Although there are other possible solutions using equipments from other manufactures, we wanted a solution that was compatible with the equipment available, in this case we have 2 Cisco Catalyst 6506 switches that support VSS so we have to choose between implementing the most used and general solution and to implement a scenery based on VSS.

We had some concerns about VSS, such as the fact that the two switches are not symmetrical since one of them has more modules installed that the other, but after consulting the VSS documentation [9] available on Cisco's website we discovered that this would not be a problem.

Our initial approach to this choice was to simulate the two sceneries using Cisco's Packet Tracer simulation software, to simulate and evaluate the two solutions. However Packet Tracer is not capable of simulating the most complex equipments and protocols making this approach impossible to pursue. We tried other network simulators but none was able to simulate VSS.

3.3.1 Solution comparison

Without the possibility to simulate VSS we resorted to the documents that we could find about VSS to try to compare it with the general solution.

Most of the documents found reported that VSS was outperforming the normal solution in terms of throughput, failover and manageability. An article from Network World about VSS, states "VSS not only delivers a 20fold improvement in failover times but also eliminates layer-2 and layer-3 redundancy protocols at the same time."[19]. In this article we can find the result of some tests run by Network World on the difference of performance, between a network with a single Cisco Catalyst 6509 switch on the distribution and core layers with a network with two 6509 switches merged with VSS, and found that the VSS-enabled virtual switch moved a record 770 million frames per second in one test, and routed more than 5.6 billion unicast and multicast flows in another. Those numbers are exactly twice what a single physical Catalyst 6509 can do.

In terms of failure recovery time that same article also contains the results of some tests where they tested the recovery time of a Layer 2/Layer 3 network, they began with a conventional setup of STP on layer 2 and HSRP at layer 3 and with 16000 hosts emulated sending traffic across redundant pairs of access, distribution and core switches, and compared the results with the same network architecture but with VSS enabled. The first setup took 6.883 seconds to recover while the second converged much faster with a record of 322 milliseconds.

At this point we can almost say with certainty that VSS is a superior solution, then we still have the fact that VSS replaces STP and redundancy protocols, since from the rest of the network the VSS virtual router is seen as one machine only enabling the creation of MECs. This makes VSS an active-active solution, this way we do not end up with blocked links and passive equipments that are only enabled when the active counterpart fails.

In figure 4.3 we can see that by enabling VSS on the distribution level we eliminate the need for STP and redundancy protocols, because it becomes seen as a single switch therefore we do not have loops in our network thanks to the MECs.

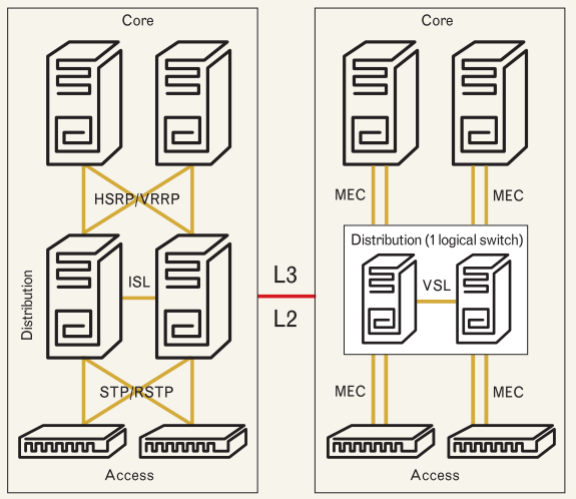


Figure 3.4: General solution versus VSS [19]

3.3.2 Implemented Solution

The core switches for this project are the Cisco Catalyst 6506. This switches are compatible with VSS enabling the use of this technology. Given the high availability and simple management characteristics of the VSS, it is the most suitable solution to use in this case. However those switches are the core of INESC TEC network, making this solution impossible to implement at this time, because as is referred on appendix A, the switches need to reboot to enable the VSS, this would mean disabling the entire network for some minutes.

So for this project we have chosen to implement a general solution using the equipment available on FEUP Netlab. The equipment available were six Cisco Catalyst 3560 switches and six Cisco 2900 series Routers.

Chapter 4

Implementation

This chapter presents the implementation of the solution.

4.1 Implementation

Using the available equipment it was possible to implement a network composed of a core and distribution layers with a redundant connection to the internet using two routers to connect to 2 different ISPs. For the core layer, since the catalyst 3560 switches have reduced routing capabilities (they can not run VRRP and GLBP like catalyst 6500 can) we have paired them with a 2901 router making the pair one of core "switches". This can be better described by the figure 4.1

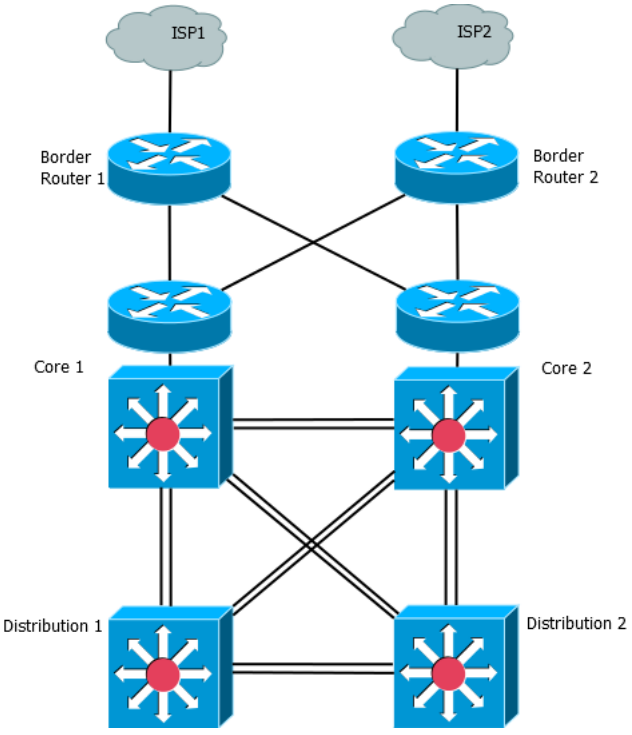


Figure 4.1: Network Diagram.

All the connections between the core and distribution layers are 2 ports link aggregations, this required Spanning Tree to be enabled on those layers to manage the redundant links. Both core routers compose a virtual router using one of the redundancy protocols mentioned before. This routers are connected to the border routers that each connect to a different ISP using BGP.

To simulate the ISPs we used two 2901 routers that simulate two different autonomous system (AS), being each one an ISP as shown on figure 4.2. Our redundant network was configured as AS10 and the ISPs were AS50 and AS60.

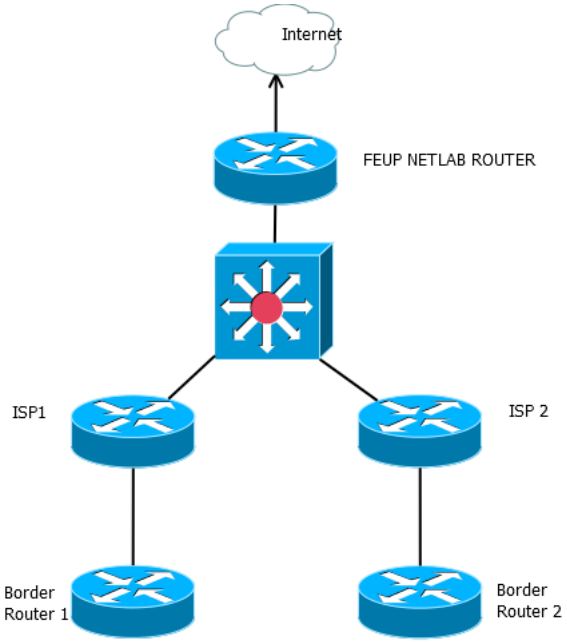


Figure 4.2: Network Diagram.

4.1.1 Network Connections and Configurations

We established six subnetworks for this implementation, one for the desired network (Home), one for the connection between our network and the border routers (Border), two for the connections between the border routers and ISP routers (Connection) and two more for the ISP Autonomous System (ISP). Table 4.1 presents the subnetworks IP addresses and netmask.

Table 4.1: Subnetworks

Subnetwork	IP address	Netmask
Home	10.0.0.0	255.255.255.0
Border	10.10.10.0	255.255.255.248
Connection 1	10.2.2.0	255.255.255.252
Conenction 2	10.3.3.0	255.255.255.252
ISP 1	10.22.22.0	255.255.255.0
ISP 2	10.33.33.0	255.255.255.0

After the initial designs we established the network from the bottom to the top.

The first step was to establish the connections between the switches, we wanted to have redundant links between them so we connected two cables between each switch on the ports described on table 4.2 and, using Cisco Etherchannel, we bonded those ports as one establishing six Port-channels between the switches.

Table 4.2: Ports used to create the Port-Channels

Port-Channel	Ports			
	Core 1	Core 2	Dist 1	Dist 2
1	23/24	23/24	-/-	-/-
2	21/22	-/-	23/24	-/-
3	-/-	21/22	-/-	23/24
4	-/-	19/20	21/22	-/-
5	19/20	-/-	-/-	21/22
6	-/-	-/-	19/20	19/20

Then we configured IP Addresses on the switches, described on table 4.3, so that we could easily access them using SSH, making the configurations easier and faster.

Table 4.3: Switches IP addresses

Switch	IP Address
Core 1	10.0.0.10
Core 2	10.0.0.40
Distro 1	10.0.0.20
Distro 2	10.0.0.50

We enabled Spanning Tree on Vlan 2, the Vlan configured as the ports access, and defined Core 1 as the Spanning tree root and Core 2 as the standby root.

After the configuration of the switches it was time to set-up the core routers. Core Router 1 was connected directly to Core Switch 1, using the GigabitEthernet interfaces available on both equipments, and Core Router 2 was connected to Core Switch 2. The Core Routers were then connected to the Border Routers using another switch. The Core and Border Routers were configured with the IP addresses presented on table 4.4.

Table 4.4: Core and Border Routers interfaces IP addresses

Router Port		IP Address
Core 1	Gigabit 0/0	10.10.10.1
	Gigabit 0/1	10.0.0.19
Core 2	Gigabit 0/0	10.10.10.2
	Gigabit 0/1	10.0.0.29
Border 1	Gigabit 0/0	10.2.2.1
	Gigabit 0/1	10.10.10.3
Border 2	Gigabit 0/0	10.3.3.1
	Gigabit 0/1	10.10.10.4
Extra Switch 1		10.10.10.6

VRRP was the redundancy protocol enabled, during the implementation, on the GigabitEthernet 0/1 interfaces of Core 1 and 2 with VRRP group 1 and the virtual IP address 10.0.0.254. The hosts connected to our network would use this IP address as their default Gateway.

At this point our network was established but some configurations were still amiss that will be presented ahead.

Now for the configuration of the fictitious ISPs, the remaining routers were used to establish two autonomous systems simulating two different ISPs. Since this two routers only have 2 interfaces available it was required to create virtual interfaces to meet our design. Each router is connected to one of the Border routers and to another extra switch. This switch has four Vlans, two of them are the ISPs own networks (Vlan 22 and 33), another is the vlan that connects the ISP routers to the NETLAB network to simulate internet access (Vlan 44) and a vlan that is used to manage this switch (Vlan 2) that connects to our network with the IP address 10.0.0.30.

The IP addresses given to the ISP routers interfaces can be visualized on table 4.5.

Table 4.5: ISP Routers interfaces IP addresses

Router interface		IP address
ISP 1	Gigabit 0/0.1	10.22.22.59
	Gigabit 0/0.2	172.16.1.59
	Gigabit 0/1	10.2.2.2
ISP 2	Gigabit 0/0.1	10.33.33.69
	Gigabit 0/0.2	172.16.1.69
	Gigabit 0/1	10.3.3.2

At this point all the equipment used was configured with the exception of BGP and Network Address Translation (NAT) that will be presented now.

At this point all used equipment was configured without BGP and Network Address Translation (NAT).

BGP was configured on all the routers, the Core and Border Routers were configured as AS 10, ISP 1 as AS 50 and ISP 2 as AS 60. The Core routers advertised the 10.0.0.0 and the 10.10.10.0 sub network and had the Border routers configured as neighbours. The Border routers advertised the 10.10.10.0 and the sub network that was used to connect to the respective ISP, in case of Border 1 10.2.2.0 and in case of Border 2 10.3.3.0, they also advertised themselves as default gateway to the Core routers, they had the respective ISP router and the Core routers as neighbours. The ISP routers advertised their respective sub networks and 172.16.1.0 and had the corresponding Border router as neighbour and the other ISP. This configuration is best verified on table 4.6.

The default Gateway of the Border and ISP routers was 172.16.1.254, the IP address of the Netlab router that connects to the exterior. Having BGP enabled and configured this way enabled two different paths to the exterior. The default Gateway of the Core routers was acquired through BGP being the possible gateways the Border routers.

Now to have connectivity to the exterior, NAT is needed to be configured on the layers that required it. The first NAT configured was on the Core routers this NAT converted the IP range of

Table 4.6: BGP configurations

Router	Core 1/2	Border 1	Border 2	ISP 1	ISP2
Advertised Nerworks	10.0.0.0 10.10.10.10	10.10.10.0 10.2.2.0	10.10.10.0 10.3.3.0	10.2.2.0 10.22.22.0 172.16.1.0	10.3.3.0 10.33.33.0 172.16.1.0
Neighbours	Border 3/4	Core 1/2 Border 2 ISP 1	Core 1/2 Border 1 ISP 2	Border 1 ISP 2	Border 2 ISP 1

the 10.0.0.0 network to the respective router IP on the network 10.10.10.0. The second layer of NAT was deployed on the Border routers, here the IP addresses of Core 1 and 2 were translated to the IP of the connection network between the Border and ISP routers. The Last layer of NAT was enabled on the ISP routers translating the IP address of the corresponding Border router and its own network range of IP to its IP in the 172.16.1.0 network in order to establish connectivity to the exterior. To note that each layer of NAT only allowed the translation of IP from the network hierarchically lower to the router were it was configured.

The commands used to configure this network can be found in appendix A section 2.

4.1.2 Hosts configurations

In order to better manage and test the network, some hosts were configured on this network. It was configured a host in each ISP with an FTP server to test the limits of the network, and four hosts on our network one of them running Nagios and MRTG to monitor the network.

Nagios is a piece of software that monitors an IT infrastructure to make sure that systems and services are working properly. MRTG is another software that, using SNMP requests, analyses the network equipment bandwidth over time.

The hosts were configured with the following IP address (table 4.7) and connected to the Distribution switches using the GigabitEthernet, when possible, and FastEthernet interfaces with the exception of one that was connected using a 2 ports bond.

Table 4.7: Host IP addresses

Host	IP	Network	Purpose
Tux-41	10.0.0.41	Home	FTP server - Tests
Tux-42	10.0.0.42	Home	FTP server - Tests
Tux-43	10.0.0.43	Home	FTP server - Tests
Tux-44	10.0.0.45	Home	Wireshark - Tests
Tux-14	10.0.0.14	Home	Nagios and MRTG
Tux-24	10.22.22.24	ISP 1	FTP server - Tests
Tux-32	10.33.33.32	ISP 2	FTP server - Tests

To create the bond we required to install the ifenslave package that enabled us to create a virtual interface, in this case Bond0, that enslaved two of the available interfaces, as shown on figure 4.3

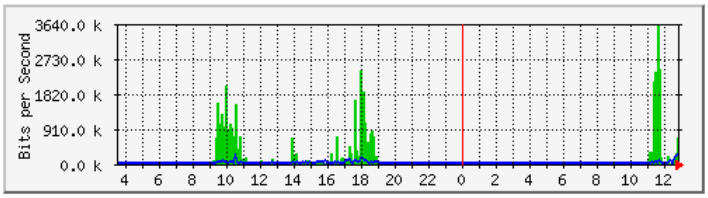
```
bond0     Link encap:Ethernet  HWaddr 00:c0:df:25:1a:f4
          inet addr:10.0.0.45  Bcast:10.0.0.255  Mask:255.255.255.0
          inet6 addr: fe80::2c0:dfff:fe25:1af4/64 Scope:Link
          UP BROADCAST RUNNING MASTER MULTICAST  MTU:1500  Metric:1
          RX packets:2931136370 errors:0 dropped:8091 overruns:0 frame:0
          TX packets:316425 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:264685958388 (246.5 GiB)  TX bytes:33822304 (32.2 MiB)

eth1      Link encap:Ethernet  HWaddr 00:c0:df:25:1a:f4
          UP BROADCAST RUNNING SLAVE MULTICAST  MTU:1500  Metric:1
          RX packets:492873 errors:0 dropped:0 overruns:0 frame:0
          TX packets:235391 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:181063660 (172.6 MiB)  TX bytes:23774088 (22.6 MiB)
          Interrupt:20 Base address:0xa000

eth2      Link encap:Ethernet  HWaddr 00:c0:df:25:1a:f4
          UP BROADCAST RUNNING SLAVE MULTICAST  MTU:1500  Metric:1
          RX packets:2930643497 errors:0 dropped:12 overruns:0 frame:0
          TX packets:81034 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:264504894728 (246.3 GiB)  TX bytes:10048216 (9.5 MiB)
          Interrupt:21 Base address:0xe100
```

Figure 4.3: Bond interface.

MRTG was configured to monitor the bandwidth of all the switches and routers interfaces. Figure 4.4 shows the daily statistics of one of the routers interfaces.

The statistics were last updated **Thursday, 27 August 2015 at 12:50**, at which time **'tux-rtr1'** had been up for **41 days, 17:36:29**.

`Daily' Graph (5 Minute Average)



	Max	Average	Current
In	3635.9 kb/s (0.4%)	121.5 kb/s (0.0%)	665.6 kb/s (0.1%)
Out	274.3 kb/s (0.0%)	14.4 kb/s (0.0%)	274.3 kb/s (0.0%)

Figure 4.4: MRTG daily statistics.

Nagios was configured to monitor all the equipments involved as shown on figure 4.5.

Linux Servers (linux-servers)

Host	Status	Services	Actions
tux14	UP	8 OK	
tux24	UP	8 OK	
tux32	UP	8 OK	
tux41	UP	8 OK	
tux42	UP	8 OK	
tux43	UP	8 OK	
tux44	UP	8 OK	

Network Routers (routers)

Host	Status	Services	Actions
tux-rtr1	UP	1 OK	
tux-rtr2	UP	1 OK	
tux-rtr3	UP	1 OK	
tux-rtr4	UP	1 OK	
tux-rtr5	UP	1 OK	
tux-rtr6	UP	1 OK	

Network Switches (switches)

Host	Status	Services	Actions
tux-sw1	UP	9 OK	
tux-sw2	UP	9 OK	
tux-sw3	UP	4 OK	
tux-sw4	UP	9 OK	
tux-sw5	UP	6 OK	
tux-sw6	UP	2 OK	

Figure 4.5: Nagios service overview.

For the hosts we configured Nagios to monitor the CPU load, latency, disk and swap usage, number of processes and the state of HTTP and SSH services as we can see in figure 4.6.

Host	Service	Status	Last Check	Duration	Attempt	Status Information
tux14	Current Load	OK	08-26-2015 15:16:55	0d 0h 24m 34s	1/4	OK - load average: 0.02, 0.27, 0.36
	Current Users	OK	08-26-2015 15:18:07	0d 0h 23m 22s	1/4	USERS OK - 6 users currently logged in
	HTTP	OK	08-26-2015 15:19:18	0d 0h 22m 11s	1/4	HTTP OK: HTTP/1.1 200 OK - 454 bytes in 0.001 second response time
	PING	OK	08-26-2015 15:20:30	0d 0h 20m 59s	1/4	PING OK - Packet loss = 0%, RTA = 0.03 ms
	Root Partition	OK	08-26-2015 15:17:03	0d 0h 24m 26s	1/4	DISK OK - free space: / 22061 MB (49% inode=90%):
	SSH	OK	08-26-2015 15:20:24	0d 0h 21m 5s	1/4	SSH OK - OpenSSH_6.0p1 Debian-4+deb7u1 (protocol 2.0)
	Swap Usage	OK	08-26-2015 15:16:59	0d 0h 24m 30s	1/4	SWAP OK - 100% free (7554 MB out of 7627 MB)
	Total Processes	OK	08-26-2015 15:18:10	0d 0h 23m 19s	1/4	PROCS OK: 94 processes with STATE = RSZDT

Figure 4.6: Nagios Host services.

For the Routers and Switches we configured Nagios to monitor the uptime and ping of each equipment as well as the bandwidth usage and the status of the most important interfaces (figure 4.7).

Host	Service	Status	Last Check	Duration	Attempt	Status Information
tux-sw1	GigabitEthernet 0/1 Link Status	OK	08-26-2015 15:18:30	0d 0h 33m 31s	1/3	SNMP OK - up(1)
	PING	OK	08-26-2015 15:19:21	41d 20h 45m 2s	1/3	PING OK - Packet loss = 0%, RTA = 2.19 ms
	Port-channel 1 Bandwidth Usage	OK	08-26-2015 15:17:50	28d 21h 43m 21s	1/3	Traffic OK - Avg. In = 400.0 B/s, Avg. Out = 455.7 KB/s
	Port-channel 1 Link Status	OK	08-26-2015 15:13:14	41d 20h 47m 49s	1/3	SNMP OK - up(1)
	Port-channel 2 Bandwidth Usage	OK	08-26-2015 15:17:05	57d 22h 12m 52s	1/3	Traffic OK - Avg. In = 319.0 B/s, Avg. Out = 455.4 KB/s
	Port-channel 2 Link Status	OK	08-26-2015 15:19:50	41d 20h 40m 20s	1/3	SNMP OK - up(1)
	Port-channel 5 Bandwidth Usage	OK	08-26-2015 15:14:40	28d 2h 46m 31s	1/3	Traffic OK - Avg. In = 456.8 KB/s, Avg. Out = 348.0 B/s
	Port-channel 5 Link Status	OK	08-26-2015 15:15:40	41d 20h 45m 19s	1/3	SNMP OK - up(1)
	Uptime	OK	08-26-2015 15:18:56	41d 20h 42m 58s	1/3	SNMP OK - Timeticks: (500707371) 57 days, 22:51:13.71

Figure 4.7: Nagios switch services.

To facilitate the tests we installed and configured vsftpd FTP server on several hosts to test the reliability of the port-channels. We enabled the anonymous user and configured the maximum tranfer rate allowed to anonymous users to 1 Gigabyte per second, in order make the equipment bandwidth the bottleneck of the network.

We installed Wireshark on Tux-44 and in a personal laptop to analyse the network traffic.

Chapter 5

Tests and results

This chapter presents tests that were performed and its results.

5.1 Tests

To fully grasp the depth of the implemented solution several tests were designed and performed. This includes the validation tests of our solution and the tests to assert the level of performance reached.

5.1.1 Validation Tests

The first tests performed were to validate our implementation. The most important point was the existence of connectivity with the outside so this was the first test, verification of connectivity with the exterior. To do so we tried to ping Google's DNS server with the IP address 8.8.8.8 with success and a traceroute to verify the desired path, as shown on figure 5.1 (for security reasons part of the traceroute is not displayed).

```
tux44:~# traceroute 8.8.8.8
traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 60 byte packets
 1  CoreRt2 (10.0.0.29)  1.388 ms  1.442 ms  1.498 ms
 2  BorderRt1 (10.10.10.39)  1.790 ms  1.834 ms  1.887 ms
 3  ISPRt1 (10.2.2.59)  2.979 ms  3.106 ms  3.150 ms
 4  firetux.netlab.fe.up.pt (172.16.1.254)  3.194 ms  3.693 ms  3.738 ms
 5                                          3.867 ms  3.911 ms  4.087 ms
 6                                          3.949 ms  4.395 ms  4.754 ms
 7                                          6.102 ms  6.117 ms  6.121 ms
 8                                          6.005 ms  4.361 ms  4.683 ms
 9                                          3.998 ms  2.476 ms  4.179 ms
10                                          9.627 ms  9.674 ms  9.136 ms
11                                          9.181 ms                        9.701 ms  9.941 ms
12                                          10.324 ms  11.250 ms  11.921 ms
13                                          12.458 ms                       12.396 ms                       12.830 ms
14  google-public-dns-a.google.com (8.8.8.8)  13.315 ms  13.361 ms  14.343 ms
```

Figure 5.1: Traceroute to 8.8.8.8.

Once verified the connectivity we successfully tried to reach all hosts, figure 5.2 shows us the result of the pings to the hosts in each of the fictitious ISPs.

```
tux44:~# ping 10.22.22.24
PING 10.22.22.24 (10.22.22.24) 56(84) bytes of data.
64 bytes from 10.22.22.24: icmp_req=1 ttl=61 time=1.09 ms
64 bytes from 10.22.22.24: icmp_req=2 ttl=61 time=0.889 ms
64 bytes from 10.22.22.24: icmp_req=3 ttl=61 time=0.868 ms
^C
--- 10.22.22.24 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 0.868/0.951/1.096/0.102 ms

tux44:~# ping 10.33.33.32
PING 10.33.33.32 (10.33.33.32) 56(84) bytes of data.
64 bytes from 10.33.33.32: icmp_req=1 ttl=61 time=1.13 ms
64 bytes from 10.33.33.32: icmp_req=2 ttl=61 time=0.968 ms
64 bytes from 10.33.33.32: icmp_req=3 ttl=61 time=0.992 ms
^C
--- 10.33.33.32 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 0.968/1.032/1.136/0.074 ms
tux44:~#
```

Figure 5.2: Pings to the ISP hosts.

At this point we were sure that all the equipments were correctly connected, which left us with the task of verifying the proper functioning of the redundancy measures implemented. To do so we disconnected the router that was serving as tux-44 gateway, during a ping request, to verify the correct functioning of the redundancy protocol in the Core routers. Figure 5.3 shows the ping request and we can see that a problem occurred between request 6 and request 18 this resulted on a loss of connectivity during approximately 11 seconds.

```
tux44:~# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_req=1 ttl=51 time=7.02 ms
64 bytes from 8.8.8.8: icmp_req=2 ttl=51 time=6.39 ms
64 bytes from 8.8.8.8: icmp_req=3 ttl=51 time=6.37 ms
64 bytes from 8.8.8.8: icmp_req=4 ttl=51 time=6.50 ms
64 bytes from 8.8.8.8: icmp_req=5 ttl=51 time=6.42 ms
64 bytes from 8.8.8.8: icmp_req=6 ttl=51 time=6.59 ms
64 bytes from 8.8.8.8: icmp_req=18 ttl=51 time=7.68 ms
64 bytes from 8.8.8.8: icmp_req=19 ttl=51 time=6.23 ms
64 bytes from 8.8.8.8: icmp_req=20 ttl=51 time=6.35 ms
64 bytes from 8.8.8.8: icmp_req=21 ttl=51 time=6.35 ms
64 bytes from 8.8.8.8: icmp_req=22 ttl=51 time=6.69 ms
^C
--- 8.8.8.8 ping statistics ---
22 packets transmitted, 11 received, 50% packet loss, time 21098ms
rtt min/avg/max/mdev = 6.236/6.603/7.683/0.406 ms
```

Figure 5.3: Ping to verify redundancy.

With the network validated we were ready to start the rest of the tests to analyse and compare the different technologies available.

5.1.2 Failover Time tests

Certain tests to measure the time that each protocol takes to recover from a failure were performed.

The procedure for each test involving a FHRP was similar, first we did a traceroute to the IP address of the virtual router created with the FHRP in question, to discover which router was forwarding packets for the virtual router, then during a ping command we disconnected that router from the rest of the network and measured the time that the protocol took to recover, calculated using the missing packets from the ping command and confirmed with Wireshark. Lastly we did another traceroute just to be sure the router forwarding packets changed.

To change the redundancy protocol enabled between the tests we used some Python scripts that are included in Appendix B.

To test the reaction time of BGP, we did something similar to the previous tests, in the case of STP tests we shutdown the root bridge to calculate the time STP took to reconverge, in the case of BGP we did a traceroute to discover which of the border router was forwarding packets to the outside and disconnected him during a ping command.

5.1.2.1 HSRP

On figure 5.4 we see that before the test the master router of our virtual router was CoreRt1 since it was the one that responded to the traceroute sent to the virtual IP address. This was the router that we disconnected during the test, and in the end of the test we can see that the Router that responds to our traceroute is CoreRt2 confirming the change of its state.

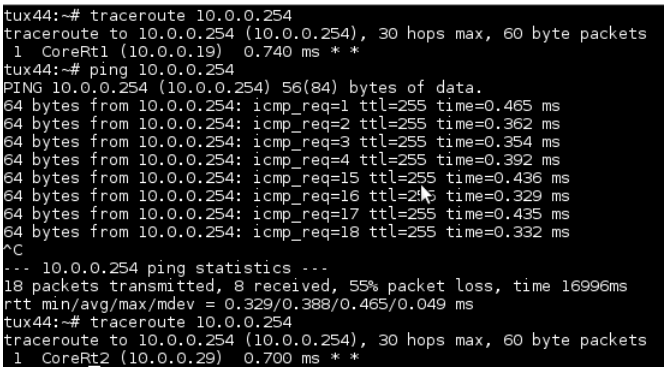


Figure 5.4: HSRP reaction time.

By counting the number of packets that were lost during the ping command we can get a approximated value of the reaction time of the protocol. In this case ten packets were lost and since the ping command sends request every second by default we can estimate that the delay is around 10 seconds. On figure 5.5 we can confirm that result by checking the Wireshark logs, we filtered the packets so that only Internet Control Message Protocol (ICMP) packets originated from the virtual router IP address were shown, and configured the time to show the amount of seconds since the last packet that is shown. We can see that packet 48 had a delay of 11 seconds (1 second of the normal ping delay plus the 10 seconds of delay) from the previous displayed packet confirming the value calculated before.

No.	Time	Source	Destination	Protocol	Length	Info
6	0.000000000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply
9	0.998898000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply
12	0.998992000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply
15	0.999041000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply
48	10.999852000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply
53	0.999893000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply
58	1.000104000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply
62	0.999897000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply

Figure 5.5: HSRP reaction time 2.

On figure 5.6 we have displayed the HSRP packets and the ICMP packets shown on the previous figure. We can see that the 11 seconds interval between packet 15 and packet 48 is the time

that CoreRt2 took to detect that the other Core was not responding and changed its state to active, making him forward the packets sent to the virtual IP address.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.0.29	224.0.0.102	HSRPv2	94	Hello (state Standby)
3	0.894102000	10.0.0.19	224.0.0.102	HSRPv2	94	Hello (state Active)
6	1.342585000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply id=0x67a2, se
7	0.227327000	10.0.0.29	224.0.0.102	HSRPv2	94	Hello (state Standby)
9	0.771571000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply id=0x67a2, se
10	0.154533000	10.0.0.19	224.0.0.102	HSRPv2	94	Hello (state Active)
12	0.844459000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply id=0x67a2, se
15	0.999041000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply id=0x67a2, se
16	0.206429000	10.0.0.29	224.0.0.102	HSRPv2	94	Hello (state Standby)
21	2.736071000	10.0.0.29	224.0.0.102	HSRPv2	94	Hello (state Standby)
28	1.903986000	10.0.0.29	224.0.0.102	HSRPv2	60	Interface State TLV (Act=0 Pass=1)
31	0.880029000	10.0.0.29	224.0.0.102	HSRPv2	94	Hello (state Standby)
36	2.568042000	10.0.0.29	224.0.0.102	HSRPv2	94	Hello (state Standby)
39	0.848115000	10.0.0.29	224.0.0.102	HSRPv2	60	Interface State TLV (Act=1 Pass=0)
40	0.000043000	10.0.0.29	224.0.0.102	HSRPv2	94	Hello (state Active)
48	1.857137000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply id=0x67a2, se
51	0.710726000	10.0.0.29	224.0.0.102	HSRPv2	94	Hello (state Active)
53	0.289167000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply id=0x67a2, se

Figure 5.6: HSRP reaction time 3.

5.1.2.2 VRRP

On figure 5.7 we see that the router that was forwarding the packets for the virtual router was CoreRt2, making this the router that we disconnected during the test. At the end of the test we confirmed that CoreRt1 switched to the role of master.

```
tux44:~# traceroute 10.0.0.254
traceroute to 10.0.0.254 (10.0.0.254), 30 hops max, 60 byte packets
 1  CoreRt2 (10.0.0.29)  0.737 ms * *
tux44:~# ping 10.0.0.254
PING 10.0.0.254 (10.0.0.254) 56(84) bytes of data.
64 bytes from 10.0.0.254: icmp_req=1 ttl=255 time=0.376 ms
64 bytes from 10.0.0.254: icmp_req=2 ttl=255 time=0.323 ms
64 bytes from 10.0.0.254: icmp_req=3 ttl=255 time=0.328 ms
64 bytes from 10.0.0.254: icmp_req=4 ttl=255 time=0.327 ms
64 bytes from 10.0.0.254: icmp_req=9 ttl=255 time=0.422 ms
64 bytes from 10.0.0.254: icmp_req=10 ttl=255 time=0.411 ms
64 bytes from 10.0.0.254: icmp_req=11 ttl=255 time=0.355 ms
64 bytes from 10.0.0.254: icmp_req=12 ttl=255 time=0.402 ms
^C
--- 10.0.0.254 ping statistics ---
12 packets transmitted, 8 received, 33% packet loss, time 10996ms
rtt min/avg/max/mdev = 0.323/0.368/0.422/0.037 ms
tux44:~# traceroute 10.0.0.254
traceroute to 10.0.0.254 (10.0.0.254), 30 hops max, 60 byte packets
 1  CoreRt1 (10.0.0.19)  0.748 ms * *
```

Figure 5.7: VRRP reaction time.

In this case we had 4 lost packets that translate roughly into a reaction of time of about 4 seconds. Analysing the packets from figure 5.8 where we applied the same filter as before, we see a delay of 5 seconds on packet 55 thus confirming the value estimated before.

No.	Time	Source	Destination	Protocol	Length	Info
26	0.000000000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply
30	0.998948000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply
34	0.999005000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply
38	0.999001000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply
55	4.999671000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply
62	0.999991000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply
67	0.999937000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply
72	1.000053000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply

Figure 5.8: VRRP reaction time 2.

Figure 5.9 shows the ICMP packets sent from 10.0.0.254 and the VRRP packets, on packet 50 we see that the source of the VRRP packets change from 10.0.0.29 to 10.0.0.19, meaning that just before this packet we disconnected the CoreRt2 and CoreRt1 not receiving VRRP packets from the master for more that 3 seconds (default timer) took the role of master and started to forward the packets sent to the virtual IP address.

No.	Time	Source	Destination	Protocol	Length	Info
24	0.855993000	10.0.0.29	224.0.0.18	VRRP	60	Announcement (v2
26	0.755729000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply
27	0.164285000	10.0.0.29	224.0.0.18	VRRP	60	Announcement (v2
30	0.834663000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply
31	0.033461000	10.0.0.29	224.0.0.18	VRRP	60	Announcement (v2
32	0.843903000	10.0.0.29	224.0.0.18	VRRP	60	Announcement (v2
34	0.121641000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply
35	0.730383000	10.0.0.29	224.0.0.18	VRRP	60	Announcement (v2
38	0.268618000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply
50	4.157901000	10.0.0.19	224.0.0.18	VRRP	60	Announcement (v2
55	0.841770000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply
56	0.050178000	10.0.0.19	224.0.0.18	VRRP	60	Announcement (v2
62	0.949813000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply

Figure 5.9: VRRP reaction time 3.

5.1.2.3 GLBP

In the case of GLBP the router that was serving our host during the tests was CoreRt1 because it was the one that the master had attributed to us, as shown on figure 5.10. We can also see a successful switch to CoreRt2.

```
tux44:~# traceroute 10.0.0.254
traceroute to 10.0.0.254 (10.0.0.254), 30 hops max, 60 byte packets
 1  CoreRt1 (10.0.0.19)  0.742 ms * *
tux44:~# ping 10.0.0.254
PING 10.0.0.254 (10.0.0.254) 56(84) bytes of data.
64 bytes from 10.0.0.254: icmp_req=1 ttl=255 time=0.395 ms
64 bytes from 10.0.0.254: icmp_req=2 ttl=255 time=0.361 ms
64 bytes from 10.0.0.254: icmp_req=3 ttl=255 time=0.345 ms
64 bytes from 10.0.0.254: icmp_req=4 ttl=255 time=0.363 ms
64 bytes from 10.0.0.254: icmp_req=15 ttl=255 time=0.385 ms
64 bytes from 10.0.0.254: icmp_req=16 ttl=255 time=0.364 ms
64 bytes from 10.0.0.254: icmp_req=17 ttl=255 time=0.335 ms
64 bytes from 10.0.0.254: icmp_req=18 ttl=255 time=0.335 ms
^C
--- 10.0.0.254 ping statistics ---
18 packets transmitted, 8 received, 55% packet loss, time 16996ms
rtt min/avg/max/mdev = 0.335/0.360/0.395/0.026 ms
tux44:~# traceroute 10.0.0.254
traceroute to 10.0.0.254 (10.0.0.254), 30 hops max, 60 byte packets
 1  CoreRt2 (10.0.0.29)  0.742 ms * *
```

Figure 5.10: GLBP reaction time.

In the case of GLBP 10 packets were lost during the recovery time which corresponds approximately to 10 seconds of reaction time for this protocol. Figure 5.11 displays the ICMP packets originated from the virtual router to our host, we can confirm the time stated before.

No.	Time	Source	Destination	Protocol	Length	Info
16	0.000000000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply
18	0.998967000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply
23	0.998986000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply
25	0.999019000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply
49	10.999563000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply
51	0.999980000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply
54	0.999972000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply
57	0.999997000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply

Figure 5.11: GLBP reaction time 2.

Note that in GLBP both the AVG and AVFs send HELLO packets to communicate that they are available, in figure 5.12 we can notice that the time between packet 25 and packet 49 is the time that CoreRt2 takes to confirm that CoreRt1 is dead, and it will start to take care of the requests sent to router 1 MAC address.

No.	Time	Source	Destination	Protocol	Length	Info
16	0.944078000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply
18	0.998967000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply
20	0.410249000	10.0.0.29	224.0.0.102	GLBP	102	G: 1, Hello, IPv4,
21	0.110724000	10.0.0.19	224.0.0.102	GLBP	102	G: 1, Hello, IPv4,
23	0.478013000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply
25	0.999019000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply
27	0.876286000	10.0.0.29	224.0.0.102	GLBP	102	G: 1, Hello, IPv4,
32	2.752070000	10.0.0.29	224.0.0.102	GLBP	102	G: 1, Hello, IPv4,
37	2.688005000	10.0.0.29	224.0.0.102	GLBP	102	G: 1, Hello, IPv4,
43	2.468018000	10.0.0.29	224.0.0.102	GLBP	102	G: 1, Hello, IPv4,
46	1.408083000	10.0.0.29	224.0.0.102	GLBP	122	G: 1, Hello, IPv4,
49	0.807101000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply
51	0.999980000	10.0.0.254	10.0.0.44	ICMP	98	Echo (ping) reply

Figure 5.12: GLBP reaction time 3.

5.1.2.4 BGP

We also tested the reaction of BGP when the border router that per default forwards the traffic to the outside gets disconnected. In this case we tweaked on the timers of BGP because the default "keep alive" timer for BGP is greater that 1 minute and we do not want our network to experience that much time without connectivity. As is shown on figure 5.13 when we sent a traceroute to the router on the other side of our "ISPs" (Netlab's router) the default path was to pass through BorderRt1 and ISPRt1. After our test the path shifted to the second ISP.

```
tux44:~# traceroute 172.16.1.254
traceroute to 172.16.1.254 (172.16.1.254), 30 hops max, 60 byte packets
 1  CoreRt2 (10.0.0.29)  1.388 ms  1.440 ms  1.497 ms
 2  BorderRt1 (10.10.10.39)  1.796 ms  1.840 ms  2.279 ms
 3  ISPRt1 (10.2.2.59)  3.075 ms  3.117 ms  3.536 ms
 4  172.16.1.254 (172.16.1.254)  3.580 ms  3.628 ms  3.670 ms
tux44:~# ping 172.16.1.254
PING 172.16.1.254 (172.16.1.254) 56(84) bytes of data.
64 bytes from 172.16.1.254: icmp_req=1 ttl=61 time=1.83 ms
64 bytes from 172.16.1.254: icmp_req=2 ttl=61 time=1.14 ms
64 bytes from 172.16.1.254: icmp_req=3 ttl=61 time=1.13 ms
64 bytes from 172.16.1.254: icmp_req=4 ttl=61 time=1.13 ms
64 bytes from 172.16.1.254: icmp_req=5 ttl=61 time=1.12 ms
64 bytes from 172.16.1.254: icmp_req=6 ttl=61 time=1.10 ms
64 bytes from 172.16.1.254: icmp_req=33 ttl=61 time=1.49 ms
64 bytes from 172.16.1.254: icmp_req=34 ttl=61 time=1.14 ms
64 bytes from 172.16.1.254: icmp_req=35 ttl=61 time=1.11 ms
^C
--- 172.16.1.254 ping statistics ---
35 packets transmitted, 9 received, 74% packet loss, time 34010ms
rtt min/avg/max/mdev = 1.104/1.248/1.837/0.237 ms
tux44:~# traceroute 172.16.1.254
traceroute to 172.16.1.254 (172.16.1.254), 30 hops max, 60 byte packets
 1  CoreRt1 (10.0.0.19)  1.393 ms  1.448 ms  1.504 ms
 2  BorderRt2 (10.10.10.49)  1.794 ms  1.839 ms  1.879 ms
 3  ISPRt2 (10.3.3.69)  3.062 ms  3.155 ms  3.553 ms
 4  172.16.1.254 (172.16.1.254)  3.605 ms  3.648 ms  3.692 ms
```

Figure 5.13: BGP reaction time.

From the packets lost we estimate a delay of 26 seconds and when compared with the time from packet 120 on Figure 5.14 we can confirm this number. Note that the keep alive time that we configured on the routers were 30 seconds so our result is within the speculated time.

No.	Time	Source	Destination	Protocol	Length	Info
24	0.000000000	172.16.1.254	10.0.0.44	ICMP	98	Echo (ping) reply
27	1.001244000	172.16.1.254	10.0.0.44	ICMP	98	Echo (ping) reply
30	1.001187000	172.16.1.254	10.0.0.44	ICMP	98	Echo (ping) reply
34	1.001179000	172.16.1.254	10.0.0.44	ICMP	98	Echo (ping) reply
37	1.001182000	172.16.1.254	10.0.0.44	ICMP	98	Echo (ping) reply
41	1.001168000	172.16.1.254	10.0.0.44	ICMP	98	Echo (ping) reply
120	27.001490000	172.16.1.254	10.0.0.44	ICMP	98	Echo (ping) reply
125	1.001197000	172.16.1.254	10.0.0.44	ICMP	98	Echo (ping) reply
127	1.001179000	172.16.1.254	10.0.0.44	ICMP	98	Echo (ping) reply

Figure 5.14: BGP reaction time 2.

In the case of BGP we can not visualize the shift as well as we can do it with the Redundancy Protocols. This is because BGP normally only sends packets when a change in the network is made so even after the change is completed several BGP packets are sent to inform the neighbours of the changes.

5.1.3 Performance tests

After the reaction time tests there was still a piece of our implementation that we had not tested and that was the link aggregation. On figure 5.15 we have the report of two simultaneous file transfers from different FTP servers(Tux 41 and 43) started from the same host (Tux 44). As we can see both transfers had a similar download speed, around 10.7 Megabytes per second, that is almost the maximum bandwidth of a FastEthernet link (100 megabits per second is equal to 12.5 Megabytes per second).

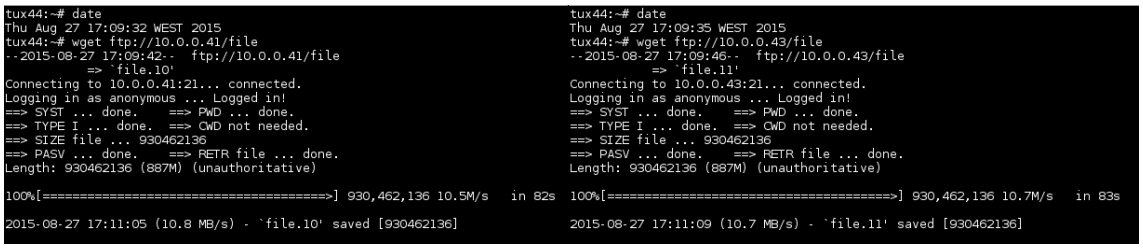


Figure 5.15: Bandwidth test.

For this test we had the two hosts with the FTP servers connected to one of the Distribution switches using FastEthernet links and our host from where we started the transfers connected to the other Distribution switch using a GigabitEthernet interface. The Distribution switches were connected using a 2 FastEthernet Interfaces Link aggregation that is estimated to have almost twice the bandwidth of a single FastEthernet Link.

We can conclude that the link aggregation was working as desired to provide resilience and a boost in performance.

5.2 Results

From the tests conducted we could validate our implementation as providing the redundancy desired and managing it as optimized as possible. The rest of the tests went as expected except for the time that HSRP took to recover from a failure, the time obtained was longer than the expected for this protocol that theoretically should have been the fastest one.

Note that the Redundancy Protocols were configured with the default configurations we did not change any of the timers, however in doing so we can better control the time that takes the protocols to recover in exchange for increasing the number o packets that travel in our network.

The main difference between the FHRP, is that in VRRP only the master router sends HELLO packets, this enables VRRP to detect a failure quicker that the other protocols since when the standby routers do not receive an HELLO package from the master during the time defined, the next router in line automatically assumes the role of master and starts to forward the packets sent to the virtual router.

In the case of HSRP when the routers become aware of the failure of the master or standby router, they start an election process to determine the next master or standby and only then the elected master starts to forward the packets sent to the virtual router.

GLBP works similar to HSRP, when the AVG or an AVF sends a HELLO packet the other members reset the timer in each they trust who sent the packet. Only when that time expires the members take action, in case of a AVG failure a new AVG is elected, normally there is already a router in standby to take the role of AVG, but when is an AVF that fails the AVG needs to distribute the hosts that were being served by that AVF between the other AVFs and elect another AVF is possible.

So if we set the hold time (default 10 seconds) to a smaller value we can reduce the reaction time for GLBP. In the case of VRRP the default HELLO time is already 1 second (and its standard does not require to support milliseconds although some switches may be able to use milliseconds) we already have the optimum time. In HSRP we can also change the hold time so that the routers declare the master to be down faster(the default is also 10 seconds like in GLBP)

Chapter 6

Conclusion and Future Work

This chapter concludes this document, with a brief conclusion of the project and some of the future work that would be relevant to conduct.

6.1 Conclusion

Reached the end of this project it is safe to conclude that there are different approaches to the subject of network redundancy.

We were subject to some unexpected problems like the fact that Cisco's Packet Tracer could not simulate newer and more complex equipment and protocols, this prevented us from simulating both the solution in order to attain more concrete data. The fact that to enable VSS the Catalyst 6500 switches needed to reboot, disrupting the proper performance of INESC TEC also prevented us from establishing a reliable network based on VSS.

Even so we were able to establish a proper redundant network using a general and independent solution based on several protocols and we were able to prove that it could withstand device and link failures.

We also found ways to automate some of the tasks involving network management, using Python scripts, reduced the time wasted on manual configuration of all the equipment.

We propose the implementation of services like OpenStack and virtualization servers and storage, since this services require high transfer rate and would enhance the user experience of INESC-TEC employees

6.2 Future Work

It would be relevant to research more technologies to establish and manage a redundant network specially those who require equipment from other manufacturers and be on the vanguard of new solutions.

It would also be relevant to enable VSS in a controlled environment to study and comprehend how the system works and the benefits of VSS as a redundancy solution.

Appendix A

Configuration Commands

This appendix contains the commands used to enable VSS and to establish the network of chapter 4.

A.1 Enabling VSS

In order to establish a VSS between the two Cisco Catalyst 6506 certain configuration are required. In this chapter a example configuration will be presented.

Step 1: Configure virtual switch ID and domain. Both switches should be configured with the same domain but different IDs

Switch 1	Switch 2
VSS-sw1#conf t	VSS-sw1#conf t
Enter configuration commands, one per line. End with CNTL/Z.	Enter configuration commands, one per line. End with CNTL/Z.
VSS-sw1(config)#switch virtual domain 100	VSS-sw1(config)#switch virtual domain 100
Domain ID 100 config will take effect only after the exec command 'switch convert mode virtual' is issued	Domain ID 100 config will take effect only after the exec command 'switch convert mode virtual' is issued
VSS-sw1(config-vs-domain)#switch 1	VSS-sw1(config-vs-domain)#switch 1
VSS-sw1(config-vs-domain)#	VSS-sw1(config-vs-domain)#

Step 2. Configure the VSL port channel and member ports. Unique port channel IDs must be chosen.

VSS-sw1#conf t	VSS-sw1(config-if)#switch virtual link 1
Enter configuration commands, one per line. End with CNTL/Z.	VSS-sw1(config-if)#no shut
VSS-sw1(config)#interface port-channel 1	VSS-sw1(config-if)#

VSS-sw2#conf t
Enter configuration commands, one per line. End with CNTL/Z.
VSS-sw2(config)#interface port-channel 2
VSS-sw2(config-if)#switch virtual link 2
VSS-sw2(config-if)#no shut
VSS-sw2(config-if)#

Now the ports of each side of the VSL should be added to the respective port channel

VSS-sw1(config)#interface range tenGigabitEthernet 5/4 - 5
VSS-sw1(config-if-range)#channel-group 1 mode on
VSS-sw1(config-if-range)#no shut
VSS-sw1(config-if-range)#^Z
VSS-sw1#

VSS-sw2(config)#interface range tenGigabitEthernet 5/4 - 5
VSS-sw2(config-if-range)#channel-group 2 mode on
VSS-sw2(config-if-range)#no shut
VSS-sw2(config-if-range)#^Z
VSS-sw2#

Step 3. Convert to virtual switch mode.

VSS-sw1#switch convert mode virtual
This command will convert all interface names to naming convention "interface-type switch-number/slot/port", save the running config to startup-config and reload the switch.
Do you want to proceed? [yes/no]: yes
Converting interface names Building configuration...
[OK]
Saving converted configurations to bootflash ...
[OK]

VSS-sw2#switch convert mode virtual
This command will convert all interface names to naming convention "interface-type switch-number/slot/port", save the running config to startup-config and reload the switch.
Do you want to proceed? [yes/no]: yes
Converting interface names Building configuration...
[OK]
Saving converted configurations to bootflash ...
[OK]

Four actions occur when this last commands is issued:

- he running configuration of the individual switch is converted into a three-level virtual switch interface notation. Two-level interface configurations (such as 10 GigabitEthernet 5/4) are converted into three-level interfaces (such as 10 GigabitEthernet 1/5/4 in Switch 1 and 10 GigabitEthernet 2/5/4 in Switch 2).
- The startup configuration is updated with the three-number notation.
- A copy of the original startup configuration converted to three-number notation is written to the multilayer switch feature card (MSFC) bootflash of the respective switch.
- Both switches reload.

When the two switches are brought online, they proceed with VSL initialization and initialize their respective VSL ports. The two switches communicate with each other and determine active and standby roles. This exchange of information is evident through the following console messages:

System detected Virtual Switch configuration...
Interface TenGigabitEthernet 1/5/4 is member of PortChannel 1
Interface TenGigabitEthernet 1/5/5 is member of PortChannel 1
<snip>
00:00:26: %VSL_BRINGUP-6-MODULE_UP: VSL module in slot 5 switch 1 brought up
Initializing as Virtual Switch active

System detected Virtual Switch configuration...
Interface TenGigabitEthernet 2/5/4 is member of PortChannel 2
Interface TenGigabitEthernet 2/5/5 is member of PortChannel 2
<snip>
00:00:26: %VSL_BRINGUP-6-MODULE_UP: VSL module in slot 5 switch 2 brought up
Initializing as Virtual Switch standby

After the VSL is initialized and the Cisco Virtual Switching System becomes active, you may notice that the console is active only for the active virtual switch and has been disabled for the standby virtual switch:

00:08:01: SW1_SP: Card inserted in Switch_number = 2,
physical slot 3, interfaces are now online
VSS >
VSS>en
VSS#

00:01:43: %CRYPTO-6-ISAKMP_ON_OFF: ISAKMP is OFF
00:01:43: %CRYPTO-6-ISAKMP_ON_OFF: ISAKMP is OFF
VSS-sdby>
Standby console disabled

Although not required, it is possible to verify that all modules have been automatically provisioned and their module types stored in the configuration by issuing the following command on the active virtual switch:

VSS#sh run | begin module provision

With the following command you can determine that the Cisco Virtual Switching System is now operating and that the two switches are acting as a single, logical network node.

VSS#show switch virtual
Switch mode : Virtual Switch
Virtual switch domain number : 100
Local switch number : 1
Local switch operational role: Virtual Switch Active
Peer switch number : 2
Peer switch operational role : Virtual Switch Standby

If the conversion process is performed using software release 12.2(33)SXI3 or newer, it is complete once the two supervisors reach the SSO Standby Hot redundancy mode. If the conversion is performed using a software release prior to 12.2(33)SXI3, there is one more critical step to perform in order to finalize the conversion.

During the conversion process, the configuration of the standby virtual switch (in this case, Switch 2) is cleared, including the configuration of the two VSL interfaces on the switch. If the switch were to reload at this point it would not have the information available to determine which interfaces to use for VSL communication. Therefore the configuration for the VSL interfaces on the standby switch must be applied, or merged from the active switch configuration. In order to facilitate this information to be repopulated again, you must complete the next step.

Step 4. Finalize the Virtual Switch Conversion

When the standby virtual switch is in SSO hot mode, you must execute the following command to automatically configure the standby virtual switch configuration on the active virtual switch:

VSS#switch accept mode virtual

This command will bring in all VSL configurations from the standby switch and populate it into the running configuration.

In addition the startup configurations will be updated with the new merged configurations.

Do you want proceed? [yes/no]: yes

Merging the standby VSL configuration. . .

Building configuration... [OK]

This concludes the initial configurations of the VSS.

A.2 Network configuration Commands

To implement the solution on FEUP Netlab several IOS commands were needed.

A.2.1 Interfaces

To change the between the access and trunk modes of the interfaces we use the following commands:

Listings A.1: Enabling Access mode

```
configure terminal
interface type number
switchport mode access
switchport access vlan vlan
end
```

Listings A.2: Enabling Trunk mode

```
configure terminal
interface type number
switchport mode trunk
switchport trun allowed vlan vlans
end
```

To configure the IP address of an interface we require this commands:

Listings A.3: Configure interface IP address

```
configure terminal
interface type number
ip address ip-address mask
end
```

To activate NAT on an interface we use this commands:

Listings A.4: Activating Nat on an interface

```
configure terminal
interface type number
ip nat inside | outside
end
```

To create the Etherchannels the following commands are required:

Listings A.5: Creating an Etherchannel

```
configure terminal
interface range type numbers
channel-group group id mode on
end
```

A.2.2 Spanning Tree Protocol

To enable STP on the switches the following commands are required:

Listings A.6: Enabling spanning tree

```
configure terminal
spanning-tree mode pvst | mst | rapid-pvst
end
```

After STP is enabled certain parameter of its behaviour can be configured at will. To change the priority of a switch:

Listings A.7: Changing the priority of a vlan

```
configure terminal
spanning-tree vlan vlan-id priority value
end
```

A.2.3 Virtual Router Redundancy Protocol

To enable VRRP:

Listings A.8: Enabling VRRP

```
configure terminal
interface type number
ip address ip-address mask
vrrp group ip ip-address
end
```

To configure VRRP timers:

Listings A.9: Configuring VRRP timers

```
configure terminal
interface type number
vrrp group timers advertise [msec] interval
end
```

A.2.4 Hot Standby Redundancy Protocol

To enable HSRP:

Listings A.10: Enabling HSRP

```
configure terminal
standby group-number ip ip-address
end
```

To configure HSRP timers:

Listings A.11: configuring HSRP timers

```
configure terminal
standby group-number timers hellotime holdtime
end
```

A.2.5 Gateway Load Balancing Protocol

To enable GLBP:

Listings A.12: Enabling GLBP

```
configure terminal
interface type number
ip address ip-address mask
glbp group ip ip-address
end
```

To configure GLBP load balancing mode:

Listings A.13: Configuring GLBP load balancing

```
configure terminal
interface type number
glbp group load-balancing host-dependent | round-robin | weighted
end
```

To configure GLBP weight:

Listings A.14: Configuring GLBP weight

```
configure terminal
interface type number
glbp group weighting maximum [lower lower] [upper upper]
end
```

To configure GLBP timers:

Listings A.15: Configuring GLBP timers

```
configure terminal
interface type number
glbp group id timers [msec] hellotime [msec] holdtime
end
```

A.2.6 Network address translation

To create and add IP address to an access list:

Listings A.16: Configuring Access Lists

```
configure terminal
access-list number permit IP address
end
```

To enable NAT:

Listings A.17: Configuring NAT

```
configure terminal
ip nat pool name start-ip end-ip prefix-length prefix-length
ip nat inside source list access-list pool name overload
```

A.2.7 Show

Some commands to verify the configuration and the state of the equipment are: Show running configuration, Show Vlan, Show IP routes, Show BGP, Show GLBP, Show VRRP, Show HSRP.

Appendix B

Python scripts

This appendix includes the Python scripts used in the tests to change the redundancy protocols.

This is the HSRP enabling script

Listings B.1: HSRP activation script

```
from netmiko import ConnectHandler
from ciscoconfparse import CiscoConfParse

r1 = {
    'device_type': 'cisco_ios',
    'ip': '10.0.0.19',
    'username': 'user',
    'password': 'pass',
}

r2 = {
    'device_type': 'cisco_ios',
    'ip': '10.0.0.29',
    'username': 'user',
    'password': 'pass',
}

router1 = ConnectHandler(**r1)
router2 = ConnectHandler(**r2)

config_commands = ['interface␣gi␣0/1',
                   'no␣glbp␣1␣ip␣10.0.0.254',
                   'no␣glbp␣1␣load-balancing',
```

```
                    'no vrrp 1 ip 10.0.0.254',
                    'standby 1 ip 10.0.0.254',
                    'standby version 2']

output = router1.send_config_set(config_commands)

print(output)

output = router2.send_config_set(config_commands)

print output
```

This is the VRRP enabling script

Listings B.2: VRRP activation script

```
from netmiko import ConnectHandler
from ciscoconfparse import CiscoConfParse

r1 = {
    'device_type': 'cisco_ios',
    'ip': '10.0.0.19',
    'username': 'user',
    'password': 'pass',
}

r2 = {
    'device_type': 'cisco_ios',
    'ip': '10.0.0.29',
    'username': 'user',
    'password': 'pass',
}

router1 = ConnectHandler(**r1)
router2 = ConnectHandler(**r2)

config_commands = ['interface␣gi␣0/1',
                   'no␣glbp␣1␣ip␣10.0.0.254',
                   'no␣glbp␣1␣load-balancing',
                   'no␣standby␣1␣ip␣10.0.0.254',
                   'no␣standby␣version␣2',
                   'vrrp␣1␣ip␣10.0.0.254']

output1 = router1.send_config_set(config_commands)

print(output1)

output2 = router2.send_config_set(config_commands)

print output2
```

This is the GLBP enabling script

Listings B.3: GLBP activation script

```
from netmiko import ConnectHandler
from ciscoconfparse import CiscoConfParse

r1 = {
    'device_type': 'cisco_ios',
    'ip': '10.0.0.19',
    'username': 'user',
    'password': 'pass',
}

r2 = {
    'device_type': 'cisco_ios',
    'ip': '10.0.0.29',
    'username': 'user',
    'password': 'pass',
}

router1 = ConnectHandler(**r1)
router2 = ConnectHandler(**r2)

config_commands = ['interface gi 0/1',
                   'no standby 1 ip 10.0.0.254',
                   'no standby version 2',
                   'no vrrp 1 ip 10.0.0.254',
                   'glbp 1 ip 10.0.0.254',
                   'glbp 1 load-balancing weighted']

output = router1.send_config_set(config_commands)

print(output)

output = router2.send_config_set(config_commands)

print output
```

References

[1] R. Salinas, *Cisco catalyst virtual switching system (vss)*. [Online]. Available: http://d2zmdbbm9feqrf.cloudfront.net/2014/eur/pdf/BRKCRS-2468.pdf.

[2] Cisco, *Understanding rapid spanning tree protocol (802.1w)*. [Online]. Available: http://www.cisco.com/c/en/us/support/docs/lan-switching/spanning-tree-protocol/24062-146.html (visited on 2015-02-13).

[3] D. Li, P. Morton, T. Li, and B. Cole, "Rfc2281 - cisco hot standby router protocol (hsrp)", pp. 1–17, 1998.

[4] Cisco, *Hot standby router protocol features and functionality*. [Online]. Available: http://www.cisco.com/c/en/us/support/docs/ip/hot-standby-router-protocol-hsrp/9234-hsrpguidetoc.html (visited on 2015-02-14).

[5] J. Pavlik, A. Komarek, V. Sobeslav, and J. Horalek, *Gateway redundancy protocols*, 2014. DOI: 10.1109/CINTI.2014.7028719.

[6] S. Nadas, "Rfc5789 - virtual router redundancy protocol (vrrp) version 3 for ipv4 and ipv6", pp. 1–40, 2010.

[7] Y. Rekhter, T. Li, and S. Hares, "Rfc4271 - a border gateway protocol 4 (bgp-4)", pp. 1–104, 2006.

[8] Cisco, *Cisco stackwise and stackwise plus technology*. [Online]. Available: http://www.cisco.com/c/en/us/products/collateral/switches/catalyst-3750-series-switches/prod%5C_white%5C_paper09186a00801b096a.html (visited on 2015-02-14).

[9] ——, *Cisco catalyst 6500 series virtual switching system*, 2012. [Online]. Available: http://www.cisco.com/c/dam/en/us/products/collateral/interfaces-modules/network-modules/white%5C_paper%5C_c11%5C_429338.pdf.

[10] Open Networking Foundation, *Software-defined networking (sdn) definition*. [Online]. Available: https://www.opennetworking.org/sdn-resources/sdn-definition (visited on 2015-04-10).

[11] J. Lim, "Software defined networking and services", 2012. [Online]. Available: http://www.packetdesign.com/resources/white-papers/sdn-white-paper.pdf.

[12] H. Kim and N. Feamster, *Improving network management with software defined networking*, 2013. DOI: 10.1109/MCOM.2013.6461195. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=%5C&arnumber=6461195.

[13] OpenStack, *Openstack*. [Online]. Available: https://www.openstack.org/software (visited on 2015-04-10).

[14] *What is openstack?* [Online]. Available: http://opensource.com/resources/what-is-openstack (visited on 2015-04-17).

[15] *Python*. [Online]. Available: https://www.python.org/ (visited on 2015-08-04).

[16] *Paramiko*. [Online]. Available: http://www.paramiko.org/.

[17] J. M. Regan, R. E. Haberman, and M. Vrana, *Link aggregation across multiple chassis*, 2008. [Online]. Available: http://www.google.com/patents/US20080181196.

[18] K. Burak, *System and method for network redundancy*, 2004. [Online]. Available: http://www.google.com/patents/US20040165525.

[19] D. Newman, *Cisco ' s virtual switch smashes throughput records*, 2008. [Online]. Available: http://www.cisco.com/c/dam/en/us/products/collateral/switches/catalyst-6500-virtual-switching-system-1440/white%5C_paper%5C_ciscos%5C_virtual%5C_switch%5C_smashes%5C_throughput%5C_records.pdf.